LA PRIMA RIVISTA EUROPEA DI SOFTWARE PER PERSONAL COMPUTER

ISSN 0392-8896

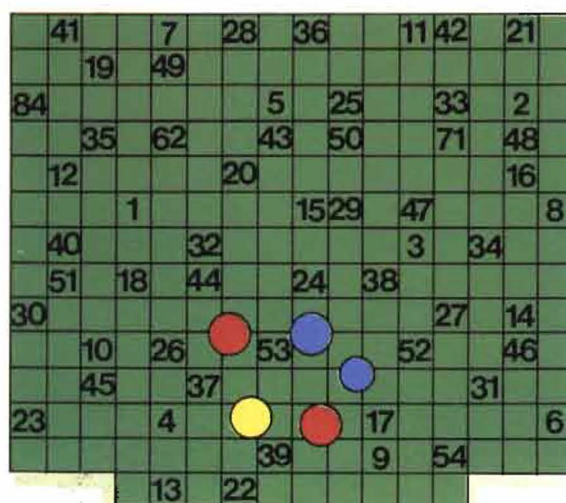
PERSONAL SOFTWARE

ANNO 3 N. 18
MAGGIO 1984 - L. 4.000

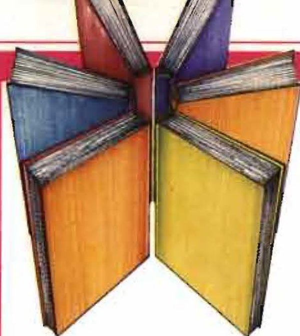
UNA PUBBLICAZIONE DEL GRUPPO EDITORIALE JACKSON



Spedizione in abb. postale Gruppo III/70



- LINGUAGGIO MACCHINA PER VIC E C 64
- TYPE-WRITER PER VIC
- CONVERTIRE SOFTWARE DA APPLE A C 64
- GRAFICA AVANZATA SUL TI99/4A
- ROBOT A CACCIA CON LO SPECTRUM
- IL C 64 IN ITALIANO
- ZX81 CROUPIER
- LA BATTAGLIA DEL LAGO GHIACCIATO CON IL PET



2+2=APPLE

Due Riviste famose, specializzate, informatissime

BIT - PERSONAL SOFTWARE

Due volumi preziosi per chi vuole approfondire la conoscenza del suo computer

INTERFACCIAMENTO DELL'APPLE
196 pagine
Cod. **334B**
Lire 14.000

APPLE II Guida all'uso
390 pagine
Cod. **331P**
Lire 26.000

Una sola firma prestigiosa per chi si interessa di informatica e di elettronica



GRUPPO EDITORIALE JACKSON

Attenzione compilare per intero la cedola
ritagliare (o fotocopiare) e spedire in busta chiusa a:
GRUPPO EDITORIALE JACKSON
Divisione Libri
Via Rosellini, 12 - 20124 Milano

COUPON D'INFORMAZIONE

Desidero ricevere un numero omaggio di □ BIT - □ PERSONAL SOFTWARE insieme a maggiori informazioni sulle condizioni di abbonamento

INVIATEMI CONTRASSEGNO

n° copie	codice	Prezzo unitario	Prezzo totale
	334B	**L. 14.000**	
	331P	**L. 26.000**	
contributo fisso spese di spedizione			L. 2000
Totale			

Nome

Cognome

Via

Cap Città Prov.

Data Firma

Spazio riservato alle Aziende. Si richiede l'emissione di fattura

Partita I.V.A.

nuovidea

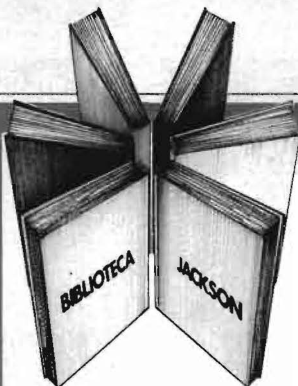


In copertina: questo mese giochiamo alla roulette con uno dei più piccoli home computer: lo ZX81.

ARTICOLI		GUIDA

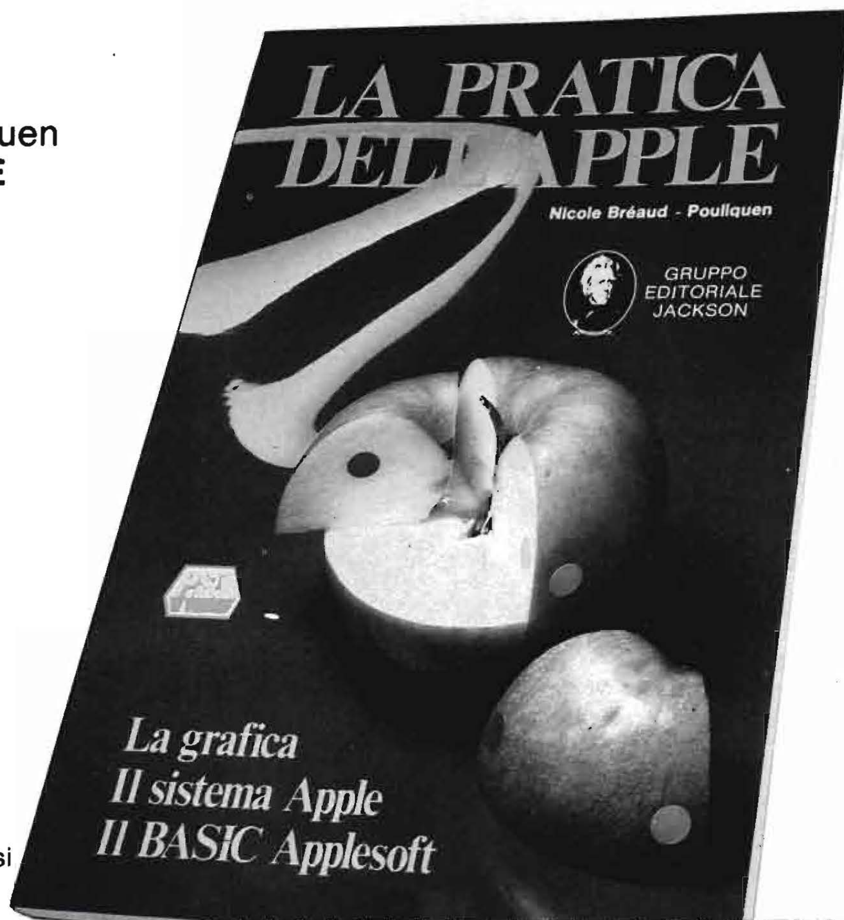
RUBRICHE		

PERSONAL SOFTWARE
N. 18
MAGGIO 1984



Personal e home computer

Provando e riprovando



Nicole Bréaud-Pouliquen
La pratica dell'APPLE

Per imparare a usare un calcolatore bisogna... usarlo. Solo così, ad esempio, è possibile scoprire e sfruttare le immense risorse operative offerte dall'**APPLE**. Provando, riprovando e... leggendo un manuale come questo.

Scritto da un vero esperto, il libro si compone di 3 capitoli fondamentali:

- **Il sistema APPLE II"** dedicato all'hardware e al software
- **"Il BASIC APPLESOFT"** con le istruzioni, i sottoprogrammi, gli operatori aritmetici e logici
- **"Il disegno e la grafica"** con le zone di memoria RAM e le funzioni grafiche.

Il tutto arricchito da numerosi esempi ed esercitazioni con soluzioni: affinchè la pratica abbia l'immediata soddisfazione del riscontro.

130 pagine
Lire 10.000
Codice **341D**



GRUPPO EDITORIALE JACKSON

Attenzione compilare per intero la cedola
ritagliare (o fotocopiare) e spedire in busta chiusa a:
GRUPPO EDITORIALE JACKSON
Divisione Libri
Via Rosellini, 12 - 20124 Milano

CEDOLA DI COMMISSIONE LIBRARIA

VOGLIATE SPEDIRMI

n° copie	codice	Prezzo unitario	Prezzo totale
	341D	**L. 10.000**	

☐ Pagherò contrassegno al postino il prezzo indicato più **L. 2000 per contributo fisso spese di spedizione.**

Condizioni di pagamento con esenzione del contributo spese di spedizione:

☐ Allego assegno della Banca ______
n° ______

☐ Allego fotocopia del versamento su c/c n. 11666203 a voi intestato

☐ Allego fotocopia di versamento su vaglia postale a voi intestato

Nome
Cognome
Via
Cap Città Prov.
Data Firma

Spazio riservato alle Aziende. Si richiede l'emissione di fattura
Partita I.V.A. ☐☐☐☐☐☐☐☐☐☐☐

nuovidea

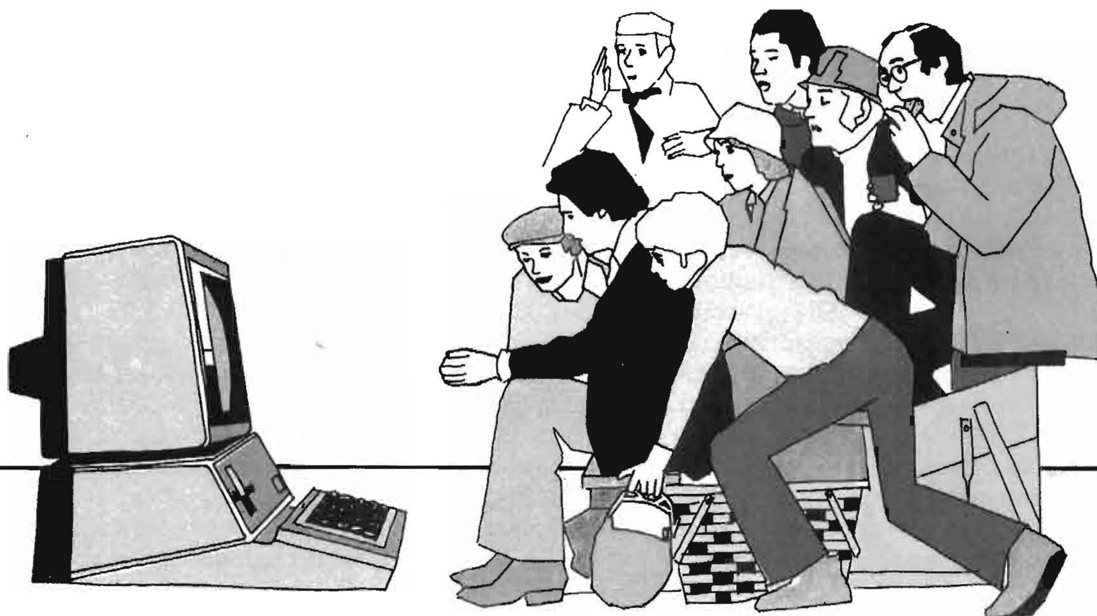
PERSONAL SOFTWARE

EDITORIALE

Le regole del gioco

di Riccardo Paolillo

I nostri lettori più affezionati e tutti quanti seguono le molteplici attività del Gruppo Editoriale Jackson, sicuramente sanno che una delle caratteristiche a denominatore comune di tutte le nostre iniziative è rappresentata dallo sforzo profuso ad una diffusione di massa dei maggiori temi informatici.
Nel corso degli ormai numerosi anni di attività, le nostre iniziative editoriali (riviste, libri, enciclopedie a dispense), hanno percorso la strada difficile, ma stimolante della divulgazione scientifica rigorosamente legata a solide basi teoriche.
In questa ottica, e coerentemente con quanto più volte affermato in questa stessa rubrica riguardo il fenomeno ormai significativo dell'informatica individuale, abbiamo sempre salutato con piacere ogni nuova iniziativa seria che potesse contribuire a diffondere l'interesse per i personal computer.
Quando però vedono la luce iniziative a carattere esclusivamente speculativo e sicuramente scorrette, ci sembra giusto operare un netto distinguo, e informarne i nostri lettori.
Realizzare e mettere in vendita, come è successo recentemente, delle cassette contenenti programmi regolarmente commercializzati nei normali negozi specializzati è una operazione tecnicamente semplice e remunerativa, ma sicuramente scorretta.
Scorretta verso chi questi programmi li ha regolarmente acquistati in negozio e ora li paga una seconda volta; scorretta, ovviamente, nei confronti di chi li ha prodotti e di chi li commercializza regolarmente a fronte di cospicui investimenti.
Ma soprattutto è un'azione dannosa per tutti gli operatori e gli utenti del settore microinformatico: è chiaro, infatti, che queste situazioni generano una comprensibile demotivazione nei migliori produttori di software a continuare il proprio lavoro.
Quando poi ci si accorge, come è accaduto a noi, che la posta dei lettori del fascicolo che accompagna una pubblicazione a cassette contiene due lettere con relative risposte riprese vistosamente da **Personal Software** e malamente cammuffate, si rimane quantomeno stupiti.
Comprendiamo che riuscire ad avere delle lettere autentiche a cui rispondere nel primo numero di una nuova rivista, sia un'impresa pressoché disperata.
Essere considerati un punto di riferimento potrebbe anche renderci un po' orgogliosi, se non prevalesse l'amarezza di constatare come anche un settore giovane, dinamico e in via di rapida espansione come il nostro, sia facile preda di operatori con pochi scrupoli. ■



POSTA

Uno pseudo-linguaggio per i nostri programmi?

Possiedo uno Spectrum da 48 Kbyte e vorrei farvi una piccola proposta: perché invece di pubblicare direttamente un programma, quando ne siete voi gli autori, non pubblicate piuttosto il procedimento, o l'algoritmo usato, insieme a uno scheletro del programma stesso scritto in un qualche pseudo linguaggio, magari ricalcando il formalismo Pascal o simili ...; i commenti al listato sono un buon inizio, ma sono appunto un inizio. Volendo spingere al limite questo, potreste pubblicare tutti i programmi sostituendo a quelle istruzioni che non sono standard (il 70% credo) delle chiamate a subroutine, sempre stile "Procedure" Pascal, di cui si pubblicherebbe poi a parte in un dizionario la codifica effettiva nei dialetti dei vari computer.

Tanto per fare un esempio, per stampare qualcosa in un punto preciso dello schermo, nel programma basterebbe scrivere:
REM PRINT alla posizione var1, var2;
mentre ad esempio in fondo alla rivista ci sarebbe la codifica effettiva di questa cosa fatta computer per computer.

Bolcioni Davide
Bologna

Uno dei fattori principali che determina la decisione di pubblicare o meno un articolo è quello relativo alla quantità e qualità di documentazione che accompagna il listato vero e proprio.
Avrà certamente notato che praticamente tutti i listati sono corredati dalla spiegazione per blocchi di linee e che spesso vengono pubblicati diagrammi di flusso e figure che dovrebbero facilitare la comprensione dell'algoritmo.

La sua proposta di utilizzare uno pseudo linguaggio comune a tutti i listati è sicuramente interessante da un punto di vista teorico, ma ci sembra piuttosto impraticabile per la nostra rivista.
Infatti l'alto numero di istruzioni non standard e il fatto che le caratteristiche grafiche e sonore varino sensibilmente da un personal all'altro, determinano una obiettiva difficoltà ad utilizzare un unico linguaggio formale.
Inoltre occorre tenere conto che questo procedimento, se da un lato favorirebbe la comprensione dei metodi utilizzati agli utenti già esperti, dall'altro complicherebbe le cose ai molti principianti che preferiscono un listato già pronto e funzionante per la propria macchina.

Infine, e questo forse è il problema maggiore, visto che gli articoli sono scritti in massima parte da lettori che collaborano alla rivista, occorrerebbe un metodo per diffondere questo ipotetico linguaggio con le difficoltà pratiche che è facile immaginare.
Comunque restiamo in attesa di ulteriori contributi e proposte su questo argomento.



Una stampante per Spectrum

Sono un vostro lettore da circa un anno, data nella quale ho acquistato un computer ZX Spectrum, di cui sono soddisfatto.
Ora dopo aver fatto un po' di esperienza con il BASIC, sto riuscendo a scrivere qualche programma, e quindi è venuta a crearsi l'esigenza di una stampante.

Ora, scartata a priori la ZX Printer, sentiti i pareri che circolano, mi sono orientato su qualcosa di più costoso: penso che con 500.000 o 600.000 lire si possa acquistare già qualcosa di discreto.
E qui viene il difficile, infatti a Torino è praticamente impossibile trovare qualcosa a meno di 1.300.000 lire.

In poche parole un utente Spectrum che desideri comperare una stampante 80 colonne ad aghi, carta comune, non termica, verso cosa si deve orientare? E che interfacce deve cercare? Poiché neanche questo i negozianti sanno con certezza.
Tra l'altro, avrei l'occasione di acquistare la stampante della Commodore la MPS 801, ma mi dicono che non è possibile in alcun modo usarla con lo Spectrum, è vero?

Campo Riccardo
Torino

Quando venne lanciato lo Spectrum, l'unica stampante che poteva essere utilizzata era la ZX Printer. Questa stampantina termica ha un costo particolarmente contenuto e consente di avere su carta anche le pagine grafiche. D'altra parte, però, è piuttosto lenta, stampa su carta stretta e facilmente deteriorabile.
Ora che lo Spectrum ha raggiunto un elevato livello di diffusione, sono state immesse sul mercato diverse interfacce seriali e parallele.

In particolare la Rebit ha reso disponibile una interfaccia universale, sia seriale RS 232 che parallela Centronics, mediante la quale è possibile collegare allo Spectrum qualunque tipo di stampante.
Per quanto riguarda la scelta della stampante, riteniamo che ne possa trovare di valide e con le caratteristiche richieste anche a cifre inferiori alle 1.300.000 lire richieste (ma un po' superiori a quelle da lei indicate).
La stampante Commodore MPS 801, va collegata direttamente al bus seriale Commodore che differisce un po' dallo standard RS 232 in quanto mancano alcuni segnali. Il collegamento dovrebbe essere teoricamente possibile, ma sicuramente poi occorrerebbe del software di gestione di non facile realizzazione.

Pertanto le consigliamo di orientarsi su stampanti che utilizzino collegamenti standard (seriale RS 232 o parallelo Centronics).

POSTA

Programmi per il PC Alphatronic

Sono un ragazzo appassionato di computer e recentemente (da Dicembre) ne posseggo uno. Il computer in questione è il nuovissimo Alphatronic PC della Triumph Adler e da allora sono disperato.

Infatti, voi ben sapete che per quanto questo computer sia in giro in Italia da circa 4 mesi, non si è visto un programma applicativo proprio dell'Alphatronic.
Ora, io vi chiedo:
— non potreste, ogni tanto fare un piccolo programma (per esempio di giochi) per rendere meno dura la vita di un quattordicenne disperato di computer?

Gian Carlo Campana
Cagliari

Il PC Alphatronic della Triumph Adler è stato presentato in Italia lo scorso mese di Settembre in occasione dello SMAU, mentre la commercializzazione ha avuto inizio qualche tempo dopo.
Come succede quasi sempre, ogni volta che viene lanciato sul mercato un nuovo personal, nei primi mesi il software viene rilasciato a ritmi molto lenti.

Questo inconveniente, che ovviamente indispettisce l'utilizzatore sempre alla caccia di nuovi programmi, è comunque in parte giustificato dal fatto che il programmatore deve dedicare un po' di tempo allo studio delle possibilità offerte dal nuovo calcolatore oltre a quello normalmente necessario per la creazione del software.
Nel caso specifico del PC Alphatronic, comunque, le cose dovrebbero essere più semplici. Infatti il nuovo personal Triumph utilizza il sistema operativo CPM che è l'unico che costituisca uno standard di fatto.

Quindi, dato che sono numerosissimi i programmi che girano sotto CPM, non dovrebbe essere difficile reperirne qualcuno che, con qualche piccola modifica, possa essere utilizzato con il PC Alphatronic.
In ogni caso è auspicabile che vengano presto resi disponibili programmi scritti appositamente per il piccolo Triumph, che possano sfruttare le interessanti caratteristiche non standard (come ad esempio la grafica).

Noi, come sempre, invitiamo chi avesse qualche programma interessante a farsi vivo con la Redazione.







La battaglia del lago ghiacciato

— Parte seconda —

Continua la nostra "battaglia" con il PET CBM

di *Umberto Giovanni Barzaghi*

Come annunciato nel numero scorso di **Personal Software**, pubblichiamo ora il listato ed i REMarks del programma "La battaglia del lago ghiacciato", il cui articolo introduttivo è apparso sul numero 17.

REMarks

0 - Inizializzazione di alcune variabili utilizzate all'interno del programma. In particolare:
GH - Fissa lo spessore minimo iniziale del ghiaccio.
VH - Indica la variabilità massima dello spessore aggiuntivo del ghiaccio.
GN, RN, GR, RR - Contengono i numeri di poke corrispondenti ai caratteri con cui vengono rappresentate nel corso del gioco le pedine che compongono i due schieramenti, in base al fatto che siano russe (RN e RR) o germaniche (GN e GR), e che le unità corrispondenti si trovino sulla terraferma (RN e GN) o sulla superficie ghiacciata del lago (RR e GR).
NR - Indica il numero delle unità che compongono la fanteria russa.
5 - Inizializzazione di altre variabili e dimensionamento dei vettori necessari all'economia del programma. In particolare:
NG - Indica il numero delle unità che compongono la cavalleria teutonica.
W$(25) - Vettore gestione cursore video.
C(20,37,2) - Contiene, per ognuna delle unità video che compongono la scacchiera di gioco (20x37, appunto), un valore indicante il numero di poke corrispondente ai caratteri che compongono la cartina (1) ed uno corrispondente allo spessore del ghiaccio in ciascun punto del lago per il quale questo valore sia significativo (2).
R(NR,4) e G(NG,4) - Contengono i dati necessari ad analizzare il comportamento di ciascuna unità che compone i due schieramenti. In particolare, la posizione di indice 1 contiene il valore di ordinata y della posizione di ciascuna unità, vale a dire l'indice di riga; la 2, il valore di ascissa x, cioè l'indice di colonna. La locazione di indice 0, contiene il valore indicante la potenza della singola unità negli scontri, mentre in 3 è contenuto il valore corrispondente al peso della singola unità, che ne determina il destino nei movimenti sul ghiaccio.
AM (9,4) e NE (9,4) - Questi due vettori contengono, per le truppe amiche e nemiche, i dati necessari a determinare l'esito dei singoli scontri. Entrambi questi vettori sono, ovviamente, dimensionati sul valore massimo teorico che il numero di truppe amiche a sostegno o nemiche da fronteggiare, all'interno della Z.O.C. (Zone of Control, zona di controllo) della unità che sostiene l'attacco, può avere.
9 - Vengono inizializzate le variabili, contenenti la potenza complessiva, negli scontri, dei due schieramenti. MG, cioè potenza complessiva della cavalleria teutonica è calcolata in ragione di un punto per ogni unità (30) e della somma dei valori iniziali indicanti la potenza negli scontri delle singole unità (15x30=450). Questo valore viene, ovviamente, modificato nel corso del combattimento. MR, calcolato con gli stessi criteri di cui sopra, è inizializzato, per la fanteria russa, a 240, vale a dire la metà del valore precedente, ma distribuito tra il doppio di unità.
12-13 - Vengono inizializzate le variabili necessarie per realizzare le scritte che compaiono nel corso della partita. Esse sono spezzate sia per ragioni di spazio (Come nel caso della scritta "I cavalieri teutonici sono stati disarcionati", che non può stare su di un'unica riga video) sia perchè vengono utilizzate in casi diversi (come la scritta "sono stati feriti" che appare sia accoppiata - e sulla stessa riga - alla scritta "i contadini russi" sia - ma su due righe che compaiono in successione, per ragioni di spazio - alla scritta "I cavalieri teutonici").
15-25 - Coordinate e ciclo di introduzione delle stesse per la fanteria russa (composta da contadini), comandata dal principe Alexander Nevskij. Le coordinate sono introdotte nell'ordine in cui le singole unità si muovono, in modo, ovviamente, da consentire degli spostamenti che non diano luogo ad ingorghi ed a situazioni di stallo.
30-45 - Come sopra per la cavalleria dell'ordine teutonico, comandata dal Gran Maestro dell'Ordine Von Balk, in persona.
Inoltre si accede alla subroutine di inizializzazione del programma.
47 - Il test tende a stabilire se si stia ricreando una partita precedentemente interrotta, nel qual caso si accede al sottoprogramma che ridisegna la cartina in base alla situazione precedente all'interruzione, quindi si salta la sezione di programma immediatamente successiva.
50-65 - Dati che consentono di creare, in sede di inizio partita, la carti-

na. I dati sono suddivisi per righe e preceduti da un numero che indica la posizione a partire dal margine sinistro, in cui il gruppo di caratteri successivo va piazzato.
100 - Dopo aver provveduto a liberare lo schermo, si provvede a leggere i dati nell'ordine opportuno ed a piazzarli sullo schermo stesso.
110-130 - Viene realizzata la cornice della scacchiera.
140-150 Per ogni coppia di coordinate della cartina, viene letto e memorizzato il numero di poke corrispondente. Nel caso in cui quest'ultimo sia diverso da 160, vale a dire carattere video in negativo, nella locazione di indice 2 corrispondente, viene memorizzato un valore fittizio indicativo (99) e si salta la successiva riga di programma, in cui viene generato casualmente, ma con un campo di variabilità dato dalle variabili GH e VH, lo spessore del ghiaccio. In 147, infine, si esamina il caso in cui il carattere corrisponda al segno meno (utilizzato nella cartina per indicare, secondo la convenzione internazionale, le zone paludose), nel qual caso la locazione di indice 2 corrispondente riceve un valore indicativo necessario per segnalare il fatto.
N.B. - Lo spessore del ghiaccio viene determinato solo nel caso in cui il carattere corrispondente alle coordinate sia uguale allo spazio vuoto in negativo, poichè solo in questo caso, qualora il ghiaccio si rompesse, nel corso della partita, potrebbe essere sostituito dal carattere prescelto per indicare il fatto, senza alterare i contorni del lago. I tratti di lago, quindi, che vengono contraddistinti da caratteri a metà bianchi e neri, in senso trasversale, sono altrettanto solidi della terraferma, e così pure per l'affluente e l'effluente del lago.
155-159 - Viene piazzato, unità per unità, l'esercito russo in base alle sue coordinate attuali; siano esse quelle iniziali o quelle in cui ciascuna unità si trovava al momento di una interruzione.
In particolare:
155 - Il numero di poke che coincide con il carattere che contraddistingue la unità in questione, viene assegnato alla variabile C1. Quindi viene operato un test per stabilire se la i-esima unità è ancora attiva (valore corrispondente alla sua potenza negli scontri, superiore o uguale a —3). Il valore negativo del test di cui sopra è dato dal fatto che le unità che rimangono intrappolate nella palude (valore uguale a —3, appunto), compaiono ugualmente sulla scacchiera, pur non potendosi muovere.
156 - Nel caso in cui la unità in questione si trovi sulla superficie del lago, la variabile C1 viene modificata in modo da confondersi con il carattere di fondo; quindi invece del carattere "pedina bianca" che contraddistingue i cavalieri teutonici, si utilizza il carattere "pedina nera" in negativo per contraddistinguere gli stessi cavalieri (che appariranno sempre bianchi) in movimento sul lago. Un discorso simmetrico vale, ovviamente per le unità russe).
157 - Display del carattere prescelto nel punto opportuno.
160-165 - Analoghe alle righe 155-159 per quanto riguarda la cavalleria dell'Ordine Teutonico.
169 - Il test si rende necessario per cedere, nel caso in cui si sia ricostruita una partita precedentemente interrotta in cui il giocatore controllava la cavalleria teutonica, il controllo alla unità cui la mossa spettava al momento dell'interruzione.
175-190 - Ciclo di gestione della CR-esima unità di fanteria russa. In particolare:
175 - Se la unità in questione è scomparsa dal tavolo di gioco o è impossibilitata a muoversi - perchè imprigionata dalle paludi -, si salta la riga immediatamente successiva.
177 - Se la unità in questione ha degli handicap pendenti, a causa di precedenti scontri, si provvede a diminuire di una unità la variabile indicante l'entità degli handicap stessi e si salta quindi la riga successiva. Con questo meccanismo viene implementato quell'effetto di rallentamento del movimento, che consente di non avere incongruenze nella velocità di spostamento, ad esempio, tra truppe russe a piedi e cavalieri teutonici, una volta disarcionati.
180 - Si accede, dopo aver opportunamente inizializzato la variabile C$, in modo da segnalare a quale dei due schieramenti si fa riferimento, alla subroutine che gestisce il comportamento delle singole unità e ad una subroutine che si occupa di verificare che nessuna delle due armate sia scomparsa dal campo da gioco.
190 - Viene aggiornato il valore dell'indice CR, con modulo pari al numero di unità che compongono l'esercito russo. Si utilizza un ciclo implicito in luogo del ciclo esplicito FOR ... NEXT, in modo da non dar luogo ad errori nel caso in cui si interrompa la elaborazione per salvare su nastro la situazione attuale di una partita.
197 - Analoga alla riga 177, per quel che riguarda lo schieramento teutonico.
195-210 - Analoghe alle 175-190 per la cavalleria teutonica.
220 - Si accede al sottoprogramma che verifica le condizioni di terminazione della partita e calcola e segnala il punteggio parziale raggiunto dai due contendenti. Quindi si ricicla, in modo da realizzare due cicli

La battaglia del lago ghiacciato

impliciti che evolvono alternativamente; permettendo così di realizzare un meccanismo che consente di muovere tutte le unità di ciascuno dei due schieramenti prima che venga analizzata la situazione che si è venuta a creare (accesso alla subroutine 4000). Inoltre, a causa della differente numerosità delle due fazioni, nello stesso arco di tempo in cui tutte le unità della fanteria di Alexander Nevskij effettuano una mossa, le unità di cavalleria del Gran Maestro Von Balk effettuano due mosse, simulando quindi una velocità di movimento doppia rispetto a quella della fanteria russa.

500-999 - Sottoprogramma di gestione del comportamento della singola unità.
In particolare:
510 - Viene esaminato il caso in cui la unità in questione appartenga allo schieramento germanico (C$="G"). In questo caso le variabili NC e RC, che contengono il numero di poke del carattere che contraddistingue l'unità su terra ferma e sul ghiaccio vengono inizializzate ai corrispondenti valori che identificano i cavalieri teutoni.
Vengono inoltre inizializzate al valore delle corrispondenti specifiche della CG-esima unità le variabili X e Y (coordinate della locazione dell'unità sulla cartina di gioco) e P (peso della unità stessa).
515 - Analoga alla riga precedente nel caso in cui la unità appartenga all'esercito russo.
520 - Viene memorizzata la vecchia posizione della pedina, tramite le sue coordinate; quindi si accede alla subroutine che mostra la posizione dell'unità presa in considerazione. Nel caso in cui la variabile C$, che contraddistingue la unità suddetta e segnala a quale dei due schieramenti appartenga, coincide con la variabile JK$, che indica quale delle due formazioni si trovi sotto il controllo del computer, si accede ad una opportuna subroutine che effettua le decisioni strategiche per la unità in questione, e si salta la riga successiva.
530 - Si accede al sottoprogramma che acquisisce le decisioni del giocatore, tramite tastiera.
540-550 - In base a quale dei due schieramenti la unità appartiene, ne viene aggiornata la posizione.
560 - Si richiama il sottoprogramma che permette di stabilire se la mossa appena compiuta ha messo la unità in grado di aggredire unità nemiche all'interno della sua Z.O.C.
1000-1999 - Subroutine di acquisizione comandi da tastiera. In particolare:
1020 - L'accesso alla subroutine 5000 ed il test sul contenuto della variabile acquisita, consente di realizzare il lampeggio della posizione attualmente occupata dalla unità, fino a che non viene premuto un tasto significativo.
1030 - Controllo di correttezza sul comando acquisito.
1035 - Nel caso in cui il tasto premuto sia lo 0 - si desidera, cioè, interrompere la partita - si richiama la subroutine che salva i dati attuali della partita su nastro.
1040 - Si richiamano i sottoprogrammi che cancellano l'ultima riga video e calcolano lo spostamento conseguente alla direzione prescelta dal giocatore.
1050 - Si accede alla subroutine che analizza lo spostamento effettuato in modo da stabilire se essa porta la unità in questione ad uscire dal tabellone di gioco od a sovrapporsi ad unità amiche o nemiche. Se la situazione è modificata in modo significativo dall'ultima mossa si interrompe ogni ulteriore analisi.
1055 - Si accede al sottoprogramma che analizza i casi in cui la mossa abbia portato l'unità a muoversi su ghiaccio (ed in questo caso si analizza se il ghiaccio si sia o no rotto) o a finire nelle paludi.
1060 - La vecchia posizione della pedina viene rimpiazzata con il carattere corrispondente della cartina sottostante. Quindi, si provvede ad invertire tra loro le due variabili, in modo che l'ultima ad essere mostrata su video sia quella effettiva, prima di effettuare il lampeggio della unità mossa nella sua nuova posizione.
3000-3150 - Sottoprogramma di inizializzazione della partita. In particolare:
3000 - Le locazioni di memoria di indice 3 delle matrici R e G, vengono inizializzate al valore iniziale del peso di ciascuna unità. L'unità di misura di questo valore non viene, volutamente, specificata, così come quella dello spessore del ghiaccio. Ciò che importa, all'interno della economia del programma, non è, infatti, il valore assoluto ma quello proporzionale dei valori suddetti. Così, il peso dei cavalieri teutonici, completi di cavalcatura e corazza è pari ad otto volte quello dei contadini russi che costituiscono le truppe a piedi di Alexander Nevskij. Da notare che, nel caso in cui i cavalieri teutonici vengano disarcionati, il loro peso scende a tre volte quello dei fanti russi (pur essendo appiedati, infatti, mantengono la loro corazzatura).
3003 - Le locazioni di memoria di indice 0 delle matrici suddette vengono inizializzate al valore di partenza della potenza di ciascuna unità negli scontri. Come si può notare, la efficacia originale delle unità germaniche è pari a cinque volte una unità di fanteria russa in piena efficienza. Nel caso in cui un cavaliere sia appiedato la sua potenza scende a 3 - pari quindi a quella di un fante -, ciò poichè, pur conservando la protezione fornita dalla corazza, essa ne diminuisce la agilità e la rapidità di movimenti. Nel caso in cui, infine, una unità sia ferita il suo valore scende di un punto e mezzo. In tutti i casi in cui venga menomata la integrità di una unità, quest'ultima viene penalizzata di due turni, che ne rallentano la velocità di avanzamento anche nei turni successivi. Nel caso in cui un cavaliere teutonico viene disarcionato, la sua velocità di avanzamento viene ridotta e sincronizzata con quella della fanteria russa. Nel solo caso in cui venga

La battaglia del lago ghiacciato



ferita una unità di cavalleria, essa salta i due turni successivi, ma non ne viene successivamente rallentato l'avanzamento.
3010-3020 - Display dell'intestazione.
3030-3050 - Viene offerta all'utente la possibilità di ricaricare una partita interrotta.
3100-3150 - Acquisizione della scelta, nel caso in cui si dia inizio ad una nuova partita, circa quale dei due eserciti si desidera controllare.
4000-4999 - Sottoprogramma di esame delle condizioni di terminazione. In particolare:
4005 - Il test consente la prosecuzione della subroutine, solo nel caso in cui la mossa tocchi alla prima unità di cavalleria tedesca, sia essa o no attiva, in modo da mostrare il punteggio ad intervalli regolari rispetto allo svolgimento dello scontro.
4010 - Si richiama la subroutine che effettua il calcolo del punteggio, quindi si crea la stringa di stampa nel caso in cui il giocatore abbia scelto di controllare la fanteria russa. Il punteggio dell'esercito controllato dal giocatore viene mostrato per primo ed il punteggio della cavalleria tedesca appare in nero su bianco.
4020 - Analogamente alla seconda parte della riga precedente viene creata la stringa di stampa nel caso in cui il giocatore controlli la cavalleria teutonica.
4030 - Display del punteggio.
4040 - Il test tende a stabilire se sono state raggiunte le condizioni di terminazione previste per la vittoria della cavalleria teutonica. Queste condizioni prevedono che il rapporto iniziale di due a uno tra i due schieramenti, per quanto riguarda la potenza negli scontri, venga portato a tre a uno (inizialmente avevo previsto, addirittura, un rapporto di quattro a uno, rivelatosi praticamente proibitivo per lo schieramento teutonico, sia sotto il controllo del computer, sia sotto il controllo del giocatore; il rapporto scelto è ancora un po' alto, se si ha sfortuna nell'attraversamento della superficie ghiacciata del lago, per questo è consigliabile cercare un rapido aggiramento del fianco destro dello schieramento russo, sfruttando la stretta striscia di terraferma tra la superficie del lago ed i confini del campo di gioco).
4050 - Test analogo a quello della riga precedente per quanto riguarda la vittoria della fanteria russa. Le condizioni, in questo caso, prevedono, semplicemente, che i due schieramenti raggiungano una sostanziale parità dal punto di vista della potenza negli scontri, poichè la cosa, dato che la minore potenza delle singole unità genera una maggior frammentazione, significa che lo schieramento russo supera numericamente l'avversario di molte unità.
5000-5010 - Mediante un ciclo, alternando i due caratteri che contraddistinguono le singole unità, in base al fatto che si trovino o no sul ghiaccio, viene realizzato l'effetto di lampeggio che ne segnala la posizione.
6000-6070 - Questa subroutine è responsabile delle decisioni sia tattiche che strategiche del calcolatore. In particolare:
6000 - Viene registrata la posizione attuale della unità sotto esame, tramite una coppia di variabili che ne conservano le coordinate, poichè le variabili originali vengono modificate nel corso dell'esame.
Viene quindi aperto un ciclo che consente di esaminare tutte le nuove alternative possibili, per quanto riguarda le mosse a disposizione del calcolatore. Le variabili X e Y, vengono ri-inizializzate all'interno del ciclo poichè esse servono da base per l'esame delle decisioni; viene, poi richiamata la subroutine che, in base alla direzione di marcia, determina la variazione delle coordinate, e si azzera la variabile DD di indice DR in cui verrà posto un valore esprimente la convenienza a muovere nella DR-esima direzione da parte della unità sotto esame.
6016-6017 - In base all'appartenenza della unità ad uno o all'altro dei due schieramenti, la variabile di indice DR viene aggiornata con un "bonus" proporzionale allo spostamento dell'unità verso lo schieramento avversario. Più semplicemente, questo meccanismo assegna un bonus positivo a quelle mosse che portano la fanteria russa a scendere verso la parte bassa dello schermo e la cavalleria tedesca a salire verso la parte alta, bonus negativi a mosse

che portino l'unità in direzioni opposte alle suddette, mentre non modificano il valore del bonus nel caso in cui lo spostamento sia esclusivamente laterale o, addirittura, (DR=5) l'alternativa esaminata è quella di non muoversi.

6019 - Viene generato, con indice opportuno, in modo da avere una serie di valori, ogni volta diversa, un valore casuale per simulare una certa "imprevedibilità" di comportamento. Alla variabile di indice DR viene sottratto un valore proporzionale allo spostamento laterale dell'unità, per evitare che, per ragioni "strategiche" che vedremo più avanti, si ottenga un movimento di ricopertura laterale dello schieramento, che porta le varie unità ad ammassarsi verso la parte destra del campo di gioco. Alla variabile suddetta vengono anche sottratti due valori proporzionali alla distanza dell'unità dal centro strategico della cartina, in modo da portare le varie unità a tendere, almeno intenzionalmente, verso questa zona dello schermo, indipendentemente da considerazioni "tattiche", che hanno comunque la preminenza.

6020 - Si determina qual'è lo stato della cartina, nella direzione in cui la DR-esima mossa porterebbe la unità in esame. Ciò serve sia ad evitare le "trappole" date dallo stato del terreno, sia eventuali problemi di coordinamento con le unità amiche. Se il terreno, nella direzione suddetta, è rappresentato da una porzione di superficie ghiacciata, il bonus viene diminuito di una unità, in modo da portare le varie unità ad evitare, se possibile, il rischio di sprofondare nel ghiaccio.

6025 - Nel caso in cui l'opzione che si esamina è quella di rimanere nel punto stesso in cui ci si trova (DR=5), se nel punto in questione la superficie non è solida ma ghiacciata (EE=160) e l'unità controllata dal calcolatore è una delle pesantissime unità di cavalleria teutonica (C$="G"), il bonus viene posto uguale ad un valore indicativo molto basso (—9999), per segnalare che la mossa in questione è pessima. Questa misura è forse un poco eccessiva, ma, dato che il passaggio sul ghiaccio ne determina un deterioramento che è proporzionale al peso della unità, sarebbe statisticamente molto rischioso rimanere fermi sul ghiaccio per una unità di cavalleria teutonica. Da notare che, nel caso in cui la mossa sia da evitare, si interrompe ogni ulteriore esame nella direzione in questione.

6030 - Mediante un ulteriore esame si sconsiglia vivamente (valore del bonus uguale a —999) di abbandonare la scacchiera.

6035 - L'esame sullo stato del terreno consente di stabilire se ci si trova in presenza di crepacci provocati da precedenti unità sprofondate nello stesso punto (EE=214); nel qual caso, la mossa viene segnalata come "sconsigliata" e l'esame interrotto.

6037 - Analoga alla 6035 nel caso di paludi (EE=45), e solo per le truppe germaniche.

6040 - Nel caso in cui lo spostamento portasse a scontrarsi con altre unità, sia amiche che nemiche, - ed il fatto viene stabilito confrontando lo stato del terreno nella direzione voluta, con i caratteri che contraddistinguono le varie unità - la mossa viene inibita nel solito modo. Da notare che dall'esame suddetto, va eliminato il caso in cui si esamini la possibilità di arrestarsi, poichè il test darebbe sempre esito positivo; dato che nella posizione suddetta si trova la unità sotto esame.

6046 - Si richiama il sottoprogramma che determina l'entità del sostegno delle unità amiche e dell'opposizione di quelle nemiche all'interno della Z.O.C. della unità sotto esame nel caso in cui si sposti nella DR-esima direzione. Nel caso in cui all'interno della zona suddetta non vi sia alcuna possibilità di opposizione avversaria, la variabile decisionale viene aggiornata aggiungendovi il valore numerico casuale calcolato in riga 6019, diminuito di .5 in modo da generare valori casuali sia positivi che negativi (la funzione RND genera, infatti, valori compresi tra 0 e 1), una opportuna frazione del numero di unità amiche all'interno della Z.O.C. (in modo da portare ad un certo effetto di concentramento delle truppe, assai utile negli scontri diretti, pur evitando quell'effetto di "ricopertura" di cui sopra) ed inoltre viene sottratto un valore PP che è proporzionale al numero di tratti paludosi che circondano la casella in cui la mossa in esame porterebbe la unità, onde evitare che una unità vada inutilmente ad infilarsi nel labirinto delle paludi che circondano il collegamento tra il lago Cudskoe ed il lato Pskov. L'esame viene quindi interrotto.

6047 - Nel caso in cui uno spostamento della unità sotto esame nella DR-esima direzione portasse ad uno scontro, si esamina la convenienza dello stesso. A questo scopo, si accede alla subroutine che stabilisce la consistenza relativa dei due schieramenti, vale a dire la somma delle variabili indicanti, appunto, la potenza negli scontri per le unità che sono coinvolte nell'eventuale scaramuccia provocata dallo spostamento nella DR-esima direzione della unità che il calcolatore sta gestendo. In base ai dati calcolati dal sottoprogramma di cui sopra, la variabile decisionale associata alla DR-esima direzione, viene aggiornata, oltre che dall'aggiunta del fattore casuale e della variabile PP, che evita, come visto alla riga precedente, che l'unità si inoltri tra le paludi; anche da un valore numerico proporzionale alla differenza tra la potenza delle truppe amiche e quella delle avversarie. In questo modo, si evita di impegnarsi pericolosamente con forze superiori, in potenza anche se non in numero, alle proprie, sia di lasciarsi sfuggire l'occasione di aggredire con un rapporto di forze vantaggioso un gruppo di unità nemiche. Ciò comporta, ovviamente, un atteggiamento piuttosto prudente, poco consono, almeno stando al film di Ejzenstejn, con quello tenuto dai cavalieri

La battaglia del lago ghiacciato

teutonici. Nel caso in cui, invece, il calcolatore controlli l'armata di Alexander Nevskij, ciò significa che il calcolatore, nell'arco di un'intero incontro, assumerà l'iniziativa di aggredire delle unità avversarie non più di un paio di volte, preferendo attendere che i cavalieri teutonici siano fortemente decimati dall'attraversamento del lago; il che mi sembra in perfetta sintonia con quanto illustrato nel film.

6050-6070 - Viene selezionata la mossa che, sulle basi delle variabili decisionali calcolate dal ciclo precedente, offre le prospettive più attraenti. Dopo di che, dopo aver ripristinato il valore originale delle variabili contenenti le coordinate della unità sotto esame, si salta ad una opportuna sezione della subroutine 1000, in modo da sfruttare gli stessi meccanismi utilizzati dalle unità controllate dal giocatore.

7000-7999 - Questo sottoprogramma esamina se il tratto di campo di gioco su cui l'unità si è trasferita presenta per la sua conformazione fisica dei rischi. In particolare:

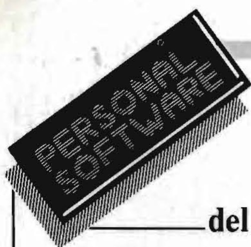
7005 - Nel caso in cui, in corrispondenza delle coordinate del punto, all'interno della matrice C di indice 2 vi sia associato un valore numerico superiore a 90, il che sta ad indicare che il tratto in questione non è ghiacciato, si salta la riga successiva.

7010 - Il valore indicante lo spessore del ghiaccio nel punto considerato viene diminuito di un valore pari alla metà del peso della unità che vi è passata, in questo modo il deterioramento del ghiaccio è proporzionale al peso delle unità che transitano sullo stesso punto. Il ghiaccio, quindi reggerà al massimo uno o due passaggi di cavalieri teutonici (con l'armatura, per di più!), mentre dei fanti russi potranno transitare anche ad interi gruppi di dieci o dodici sullo stesso punto prima di provocare dei crepacci.

7020 - Nel caso in cui la variabile C di indice 2, corrispondente alle coordinate del punto in cui si trova la unità in questione, ha un valore di 99, segnala cioè la presenza di terraferma, l'analisi viene interrotta.

7030 - Nel caso in cui il valore numerico associato sia di 90 (=palude), si richiama la subroutine che segnala l'incidente capitato alla unità suddetta, solo se questa è teutonica.

7040 - Nel caso in cui il peso della unità in transito sulla casella in esame del campo di gioco, sia superiore al valore numerico associato alla casella stessa, ed esprimente lo spessore del ghiaccio, si accede al sottoprogramma che segnala la rottura del



La battaglia del lago ghiacciato

ghiaccio stesso.

8000-8999 - Sottoprogramma di controllo degli scontri. In particolare:

8000 - Il test consente di interrompere l'esame della situazione nel caso in cui la scomparsa dell'unità in seguito alla sua ultima mossa (per essere stata inghiottita dal lago ghiacciato) abbia reso l'indagine inutile.

8010 - Si accede al sottoprogramma che determina il numero delle unità, sia amiche che nemiche, all'interno della zona di controllo dell'unità che ha mosso per ultima.

8110 - Nel caso in cui non ci sono truppe nemiche a distanza tale da consentire uno scontro, non è necessario proseguire nell'esame della situazione.

8120 - Si richiama la subroutine che determina l'entità del rapporto di forze tra i due schieramenti (composti anche solo di una unità per parte) che partecipano allo scontro.

8130 - In base ai dati calcolati dalla subroutine di cui sopra, vengono determinati dei valori percentuali proporzionali alla potenza dei due schieramenti che servono da limite per determinare la C.R.T. (Combat results table, tabella dei risultati negli scontri), che determina l'esito degli scontri stessi. Il limite inferiore spetta, ovviamente, al più forte dei due schieramenti (come potenza, non come numero), in modo da rendere inferiori le sue probabilità percentuali di subire danni nello scontro. Viene inoltre generato casualmente un valore compreso tra 0 e 100, diretto responsabile dell'esito dello scontro.

8140 - Nel caso in cui il valore numerico casuale sia inferiore ad un terzo del limite percentuale associato allo schieramento amico, si accede al sottoprogramma che segnala ed opera l'eliminazione di tutte le unità amiche coinvolte nello scontro.

8150 - Nel caso in cui il valore generato casualmente sia compreso tra un terzo e due terzi del valore limite, si accede alla subroutine che determina, nel caso delle unità russe il loro ferimento, e nel caso dei cavalieri teutonici, il loro disarcionamento.

8154 - Nel caso in cui il valore suddetto sia compreso tra due terzi ed il valore limite intero, spettante alle unità amiche, si accede alla subroutine che, per entrambi gli schieramenti, ne designa il ferimento.

8156 - Se il valore random è compreso tra il limite per le unità amiche ed il limite suddetto più un terzo del restante valore percentuale si accede ad una subroutine analoga a quella di riga 8154, ma riguardante le unità nemiche. La distinzione si rende necessaria a causa del meccanismo di registrazione delle unità che compongono i due schieramenti coinvolti nella scaramuccia.

8160 - Analoga alla riga 8150; in questo caso, però, vengono coinvolte le truppe avversarie.

8170 - Analoga alla riga 8140, come sopra, però, non vengono rimosse le unità amiche, ma quelle avversarie.

8500-8510 - Questa subroutine è direttamente responsabile della registrazione dei dati delle unità amiche coinvolte in uno scontro.

In particolare:

8500 - Nella matrice AM di indice AM (valore attuale del numero di unità amiche all'interno della Z.O.C.) vengono registrate le coordinate dell'unità (indice 1 e 2).

Quindi si accede ad una subroutine che determina il numero di serie della unità coinvolta in base alla sua posizione (che è determinata univocamente, poichè non sono possibili sovrapposizioni di unità diverse), in modo da poter determinare gli altri dati della unità.

Nella matrice AM di indice 0, viene registrato il numero di serie progressivo suddetto (che rappresenta l'ordine in cui le varie unità hanno diritto a muovere). Nella locazione di indice 3, viene invece registrato il valore numerico esprimente il peso della unità.

8510 - Nella locazione di indice 4 viene registrata la potenza attuale della unità negli scontri.

8600-8610 - Analoga alle righe 8500-8510, per lo schieramento avversario.

9000 - Subroutine di clearing della vecchia posizione occupata dalla ultima unità che si sia mossa.

10500-10590 - Sottoprogramma di caricamento dati da nastro, per consentire la prosecuzione di una partita precedentemente interrotta.

10900-10940 - Questa subroutine permette di ricostruire il campo di gara, sia come cornice che come cartina, nella situazione in cui si trovava precedentemente alla interruzione.

11000-11999 - Subroutine di salvataggio su nastro di una situazione intermedia. In particolare:

11010 - Viene salvata, in una variabile MC$, la variabile C$, indicante a quale dei due schieramenti la mossa spetta, in modo da, non commettere errori di attribuzione della mossa, al momento di riprendere una partita interrotta.

12000-12020 - Questo sottoprogramma provvede a segnalare al giocatore il fatto che una o più unità siano state inghiottite dalla superficie ghiacciata del lago. In particolare:

12000 - Viene azzerata la locazione di indice 2 associata alla zona del campo di battaglia in cui si sono creati dei crepacci. In questo modo, lo spessore del ghiaccio viene azzerato, quindi la casella in questione non è più in grado di sostenere il peso di una qualsiasi unità, per piccolo che esso sia.

12005 - Alla locazione di memoria di indice 1, associata alla casella, viene assegnato il valore corrispondente al numero di poke del carattere che contraddistingue il crepaccio creatosi.

12020 - Si accede, con un valore opportuno della variabile Q, ad una subroutine che provvede ad aggiornare i valori di potenza negli scontri per la unità scomparsa, oltre che i valori esprimenti il punteggio parziale della partita.

13000 - Subroutine di temporizza-

zione delle scritte.
14000-14020 - Questo sottoprogramma ha funzioni analoghe a quelle assegnate alla subroutine 12000, in questo caso, però, viene segnalato che la unità che ha appena effettuato una mossa, è rimasta intrappolata in una zona paludosa. In particolare:
14000 - La locazione di indice 1 associata alla zona di campo di battaglia in questione, viene aggiornata con il valore che contraddistingue in notazione normale (NC), la unità intrappolata. In questo modo, nel caso in cui la battaglia venga interrotta e salvata su nastro, la unità intrappolata, verrà correttamente riportata sulla cartina, così come i crepacci provocati da precedenti scomparse.
14020 - Analoga alla 12020.
15000 - Display della zona di campo di battaglia precedentemente "coperta" dall'ultima unità mossa.
16000-16999 - Subroutine di aggiornamento del valore di potenza negli scontri per una unità che venga, da un qualsiasi accidente, "tolta" dal gioco.
17000-17010 - Questo sottoprogramma esamina i casi in cui lo spostamento attribuito dal calcolatore, o quello scelto dal giocatore, per la unità cui tocca muovere, determinino l'uscita della stessa dal campo di gioco o siano impossibilitati per la presenza, nello stesso punto di altre unità. In particolare:
17000 - Nel caso in cui si esca dal campo di gioco, si accede ad una opportuna subroutine che segnala la cosa.
17005 - Viene acquisito il carattere presente nella casella di "destinazione" della unità in questione, nel caso in cui il carattere suddetto coincida con quelli che contraddistinguono le unità sul campo di gioco, le coordinate della unità vengono poste uguale a quelle di "partenza".
18000-18020 - Questo sottoprogramma segnala la "fuga" di una unità dal campo di gioco. In particolare:
18000 - Viene cancellata la vecchia posizione della unità fuggita, rimpiazzandola con il carattere che essa copriva sulla cartina.
18020 - Analoga alle righe 12020 e 14020.
19000-19020 - Subroutine di servizio per la determinazione dei caratteri che contraddistinguono le unità nemiche, in base a quelli che contraddistinguono le amiche.
20000-20540 - Questa subroutine, per così dire di "ricerca associativa", determina l'indirizzo della unità coinvolta in uno scontro - ed in base a questo indirizzo, altri suoi dati -, partendo dall'unica cosa che sia nota, da una ricerca "sul campo", cioè la coppia di coordinate che la contraddistinguono. In particolare:
20000 - Inizializzazione dell'indice W di ricerca e test per stabilire se la unità ricercata appartenga all'uno o all'altro dei due schieramenti.
20020-20040 - Ricerca tra lo schieramento russo.
20500-20540 - Ricerca tra lo schieramento teutonico.
21000-21020 - Altra subroutine di servizio che determina qual'è la variabile "stringa" che contraddistingue le forze avversarie, partendo da quella che identifica le amiche.
22000-22010 - Mediante un ciclo, vengono rimosse, accedendo ad una opportuna subroutine, che provvede anche ad aggiornare le variabili associate, le unità amiche che hanno avuto la peggio in uno scontro.
23000-23540 - Questo sottoprogramma provvede a comunicare il ferimento delle unità russe (nel caso in cui queste siano controllate dallo schieramento, la cui ultima mossa ha determinato lo scontro) od il disarcionamento, nel caso in cui le unità siano teutoniche. In particolare:
23010 - Nel caso in cui la unità in questione sia russa, si salta ad una opportuna sezione della subroutine.
23020-23040 - Questa sezione di subroutine si occupa del disarcionamento delle truppe tedesche. In particolare:
23020 - Viene aperto il ciclo che consente di stabilire la sorte degli AM contingenti che partecipano allo scontro. Viene inoltre determinato l'indice IN, corrispondente al numero d'ordine della unità. Nel caso in cui il peso della unità di cavalleria in questione sia già pari a tre unità (il che significa che l'unità stessa è già stata disarcionata) si richiama il sottoprogramma che comunica il ferimento della unità stessa. In quest'ultimo caso il ciclo viene aggiornato, in questo modo, nel caso in cui nello scontro siano coinvolte più unità, che hanno avuto sorte differente in scontri precedenti, si possono anche avere esiti differenti per le varie unità che costituiscono il gruppo impegnato nello scontro.
23025 - Vengono aggiornate alcune variabili in base all'ipotesi che l'unità sia stata disarcionata.
23027 - La variabile G di indice 0 associata all'unità in questione viene diminuita di dodici unità, in questo modo vengono disarcionati sia cavalieri fino a quel momento illesi, sia cavalieri che, pur rimanendo in sella, siano stati precedentemente feriti. Viene inoltre aggiornata la variabile di indice 4 che contiene gli handicap dovuti a ferimenti o, appunto, disarcionamento, e che consente di rallentare il procedere di unità che siano state ferite o, come in questo caso, appiedate.
23030 - Vengono mostrate le scritte che comunicano al giocatore l'accaduto.
23500 - Analoga alla 23020 nel caso del ferimento delle unità russe. In questo caso, se le unità in questione sono già state ferite (la qual cosa è rappresentata dal valore che esprime la potenza negli scontri eguagliato a 1.5), si accede alla subroutine che segnala la "morte" dell'unità.
23510-23535 - Analoghe alle 23025-23030.
24000-24540 - Rappresenta l'esatto simmetrico delle righe 23000-23540, la sola differenza è rappresentata dal fatto che in questo caso ad avere la peggio sono le unità avversarie di

quella che ha, con la sua mossa, provocato lo scontro.

25000-25010 - Subroutine di servizio, consente l'aggiornamento della situazione della singola unità, oltre che del punteggio generale, in seguito agli scontri.

26000 - Questo sottoprogramma consente di segnalare il disarcionamento di una unità diminuendone il valore che ne esprime il peso.

27000 - Questo sottoprogramma fa lampeggiare le unità che hanno avuto la peggio negli scontri.

28000-28020 - Questa subroutine provvede, con modalità analoghe a quelle viste in 23000, a rimuovere dal campo di gioco le unità "morte" e ad aggiornare opportunamente le variabili delle unità stesse in modo da escluderle dal corso della partita. Questa subroutine viene attivata nel caso in cui perisca una delle unità "amiche".

29000-29020 - Analoga alla precedente, nel caso in cui a "morire" siano i "nemici".

30000-30090 - Questo sottoprogramma, in base alla direzione che il calcolatore od il giocatore ha scelto per la unità cui spetta la mossa, determina l'incremento delle coordinate della unità stessa.

31000-31110 - Questa subroutine provvede a calcolare la numerosità dei due schieramenti coinvolti in uno scontro. In particolare:

31010 - Vengono azzerate le variabili AM e NE, che sono destinate a contenere il numero di unità che compongono i due schieramenti. Viene quindi aperto il ciclo che consente di esaminare le nove caselle disposte a quadrato di lato tre attorno alla pedina che per ultima si è mossa.

31020 - Viene acquisito il carattere della casella di coordinate Y+DY e X+DX. Nel caso in cui il carattere in questione coincida con uno dei due caratteri che contraddistinguono le unità dello stesso schieramento di quella che ha effettuato la mossa, viene aggiornata la variabile AM e si richiama una opportuna subroutine che provvede a registrare i dati essenziali della unità in questione, in modo da poterli utilizzare negli scontri.

31030 - Analoga alla precedente nel caso di una unità nemica.

31040 - Questa riga è fondamentalmente avulsa dal resto del contesto ed è stata inserita in questo sottoprogramma semplicemente per comodità e per questioni di rapidità (in questo modo non è necessario ripetere il ciclo all'interno del quale il test viene eseguito in altro ambiente). Viene infatti aggiornata la variabile che segnala la presenza di zone paludose attorno alla casella occupata dall'unità che ha effettuato l'ultima mossa.

32000-32999 - Questo sottoprogramma calcola la potenza dei due schieramenti che si confrontano in uno scontro, in base ai dati di potenza delle singole unità, così come sono stati registrati dalla subroutine precedente.

In particolare:

32000 - Azzeramento delle variabili che contengono i valori totali di potenza dei due schieramenti.

32010 - Ciclo di calcolo della potenza delle forze "amiche" rispetto all'ultima pedina che si è mossa.

32020 - Ciclo di calcolo delle forze totali "nemiche".

33500-33540 - Questo sottoprogramma viene richiamato nel caso in cui l'esito della subroutine 8000, sia stato di "ferimento" per le truppe che costituiscono il gruppo di supporto nello scontro per l'ultima pedina giocata. In particolare:

33500 - Nel caso in cui il cavaliere teutonico in questione sia già stato ferito in precedenza, si accede alla subroutine che rimuove la pedina corrispondente dalla scacchiera di gioco, in quanto "morta".

33510-33537 - In tutto e per tutto uguali alle 23510-23535. L'unica differenza è rappresentata dal fatto che, in questo caso, ad essere "ferite" sono le unità di cavalleria teutoniche.

34500-34540 - Analoghe alle righe 33500-33540, la differenza è data dall'appartenenza delle truppe ferite allo schieramento cui appartengono gli avversari della unità che, con la sua ultima mossa, ha provocato lo scontro che ha portato al ferimento delle truppe stesse.

35000-35020 - Questa subroutine di servizio si occupa di calcolare il punteggio parziale della partita. In particolare:

35000-35010 - I due cicli provvedono a sommare i valori di potenza negli scontri di tutte le unità che, per i due schieramenti, sono ancora attive, vale a dire hanno valori della variabile R o G di indice zero positivi (i test nei due cicli hanno appunto lo scopo di escludere i valori negativi che segnalano la fuga o la scomparsa dell'unità dal campo di gioco, perchè inghiottita dal lago ghiacciato o intrappolata dalla palude).

35020 - Ai valori RP e GP, che rappresentano la somma delle potenze negli scontri per i due schieramenti, calcolate con le modalità di cui sopra, vengono sommati i valori PR e PG, aggiornati direttamente nel corso del programma, che rappresentano il numero di unità ancora attive per i due schieramenti.

I valori così ottenuti, vengono quindi sottratti dai punteggi iniziali che, calcolati con criteri analoghi ai suddetti, rappresentano il rapporto di forza di partenza, esistente tra i due eserciti.

I punteggi così ottenuti vengono quindi utilizzati dalla subroutine che si occupa di mostrarli al giocatore, poichè rappresentano il numero che, espresso in base ai concetti di potenza negli scontri più volte visti, indica le perdite che i due schieramenti si sono reciprocamente inflitti fino a quel momento, nel corso del gioco.

36000-36999 - Questo sottoprogramma, comunica la vittoria della cavalleria teutonica.

37000-37999 - Subroutine di funzione analoga alla precedente, in questo caso la vittoria spetta alla fanteria russa.

La battaglia del lago ghiacciato

38000-38020 - Questo sottoprogramma, che viene richiamato dopo ogni mossa, serve a verificare l'ipotesi che uno o l'altro dei due schieramenti non abbia più unità attive, pur non avendo raggiunto altrimenti le condizioni di terminazione. Ciò si rende necessario poichè l'analisi della situazione, per questioni di congruenza logica, viene compiuta solo al termine dell'intero ciclo di gioco dei due schieramenti (non avrebbe affatto senso dichiarare uno dei due eserciti vincitore, se lo schieramento sconfitto ha ancora delle unità, cui toccherebbe muovere nel seguito della partita per completare il turno, e che potrebbero capovolgere la situazione, coinvolgendo le unità avversarie in nuovi scontri). È evidente che, nel caso supposto nel test di cui sopra, non avrebbe senso proseguire la partita fino al termine del turno di gioco dello schieramento superstite, poichè questo non avrebbe più avversari. In particolare:

38000-38010 - Nel caso in cui l'armata russa non abbia più unità in campo viene decretata (tramite la opportuna subroutine) la vittoria dell'esercito teutonico e viceversa.

38020 - Nel caso in cui il gioco prosegue regolarmente, si richiama la subroutine che provvede a "ripristinare" gli handicap eventualmente azzerati dall'ultima mossa.

39000-39020 - Questo sottoprogramma si occupa di comunicare ed aggiornare opportunamente le variabili coinvolte nel "ferimento" delle truppe che sostengono l'unità che ha provocato lo scontro. Da notare che, nel caso delle truppe russe, una simile subroutine già esiste (23500-23540), accoppiata però, nel caso delle truppe germaniche, alla subroutine che si occupa del disarcionamento di quest'ultime. Questo sdoppiamento ha ragioni logiche da ricercarsi nella necessità di simmetria delle subroutine che decidono i vari esiti dello scontro.

La stessa subroutine di controllo (8000) viene infatti richiamata da entrambi gli schieramenti, siano essi russi o tedeschi, a controllo del computer o del giocatore, nel momento in cui si provoca lo scontro. Supponiamo che lo scontro venga affrontato con un rapporto di 15 a 3 per lo schieramento tedesco; in questo caso il rapporto percentuale è di 83,33 probabilità su cento di esiti favorevoli ai cavalieri dell'Ordine teutonico, suddiviso tra un terzo di probabilità che l'esito sia di "morte" delle unità nemiche e due terzi che comportano il ferimento delle stesse (prescindendo dalla loro attuale situazione); di questi due terzi, però, un terzo passerebbe tramite la subroutine 23000 ed un terzo tramite la 39000. Nel caso dell'esito favorevole alle truppe russe (rapporto percentuale del 16,67%), le probabilità suddette sono equamente suddivise tra quelle che vedono la morte dei cavalieri coinvolti nello scontro, quelle che ne vedono il ferimento e quelle che vedono il disarcionamento degli stessi. In particolare:

39010 - Nel caso di truppe russe si richiama, appunto, la subroutine che svolge già, in altro ambiente, la stessa funzione.

39020 - Nel caso di truppe tedesche, è invece necessario aprire il ciclo tra le varie truppe che costituiscono il gruppo coinvolto nello scontro; calcolando gli indici opportuni in maniera non differente da quanto fatto in 23020. Quindi si richiama la subroutine utilizzata al medesimo scopo da altri sottoprogrammi.

40000-40020 - Analoga alla subroutine precedente nel caso in cui ad essere ferite siano state le truppe avversarie di quelle che hanno provocato la scaramuccia.

41000-41510 - Questo sottoprogramma, richiamato dopo ogni mossa, provvede a ripristinare gli handicap che consentono di rallentare i movimenti di quelle unità che sono state menomate (sia perchè ferite sia, nel caso dei soli cavalieri tedeschi, perchè private delle loro cavalcature). In particolare:

41000 - Se l'ultima unità mossa è russa si richiama la opportuna sezione di programma.

41010 - Se il valore che esprime il peso della unità teutonica è superiore a tre (il che sta ad indicare che il cavaliere è ancora "in sella"), si interrompe l'esame.

41020 - Se il valore che esprime la potenza della unità è superiore a 1.5, il che significa che l'unità, pur essendo appiedata non è ferita, la variabile G di indice 4, che contiene l'handicap attribuito alla unità, viene reinizializzata ad 1, in modo da eguagliare la velocità di avanzamento del cavaliere tedesco disarcionato a quella delle truppe a piedi russe, vale a dire, la metà della velocità delle truppe teutoniche a cavallo.

41030 - Nel caso in cui il valore suddetto fosse inferiore a 1.5 (l'unità fosse cioè, oltre che appiedata, anche ferita), l'handicap è pari a due. In questo modo le unità ferite e appiedate risultano più lente di quelle, pure a piedi, ma illese.

41500 - Le unità russe illese (valore di potenza negli scontri superiori ad 1.5) vengono escluse dall'esame. Da notare che la loro velocità è già pari alla metà di quella delle unità tedesche a cavallo, a causa del loro numero che le costringe ad effettuare una sola mossa nel ciclo che consente alle unità teutoniche, presenti in minor numero, di compierne due.

41510 - analoga alla 41030.

59000-59999 - Questa subroutine è del tutto inutile nella, attuale, economia del programma. Essa può essere considerata una forma di reperto archeologico dell'era del computer. È infatti servita in sede di debug del programma per rendersi conto del funzionamento del meccanismo logico che portava il computer a scegliere le proprie mosse; e può, pertanto, non essere inserita nel programma. Chi volesse vederne gli effetti, però, dovrebbe solo (oltre che, naturalmente, trascriverla) inserire una GOSUB59000, prima del NEXT in riga 6050.

In questo caso, quando la mossa spetta al calcolatore, comparirebbe-

ro, sull'ultima riga dello schermo:
— il numero del tastierino numerico associato alla direzione che il calcolatore sta prendendo in esame.
— il valore che esprime la convenienza a muoversi in quella direzione.
— il numero di truppe amiche e nemiche all'interno della Z.O.C. nella nuova posizione.
— i valori di potenza negli scontri per i due schieramenti suddetti.
— il valore random.

N.B. - per consentire una buona lettura del debug, è necessario premere un qualsiasi tasto per far si che il computer passi ad esaminare la successiva alternativa.

Listato 1. *Programma "Battaglia del lago ghiacciato".*

```
 0 R$="▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒":GH=7.
5:VH=5.5:GN=81:RN=87:GR=215:RR=209:NR=6
0
 5 NG=30:DIMW$(25),A(20,4),C(20,37,2),R
(NR,4),G(NG,4),AM(9,4),NE(9,4)
 9 MG=480:MR=240:CD=0
 10 FORI=1TO25:W$(I)=LEFT$(R$,I):NEXT:T
P(0)=1:TP(1)=-1:R6=NR:G6=NG
 12 TY$(1)="I CAVALIERI TEUTONICI":TY$(
2)="SONO STATI DISARCIONATI"
 13 YT$(1)="I CONTADINI RUSSI":YT$(2)="
SONO STATI FERITI"
 15 REM DATA ALEXANDER NEVSKIJ
 16 DATA2,10,3,10,4,10,5,10,6.10,10,7,1
1,7,12,7,13,7,14,7
 17 DATA18,5,19,5,20,5,21,5,22,5,2,9,3,
9,4,9,5,9,6,9
 18 DATA10,6,11,6,12,6,13,6,14,6,18,4,1
9,4,20,4,21,4,22,4,2,7,3,7,4,7,5,7,6,7
 19 DATA10,4,11,4,12,4,13,4,14,4,24,5,2
5,5,26,5,27,5,28,5,2,5,3,5,4,5,5,5,6,5
 20 DATA32,2,34,2,35,2,34,1,35,1,18,2,1
9,2,20,2,21,2,22,2
 25 FORI=1TONR:READR(I,2):READR(I,1):NE
XT
 30 REM DATA GROSSEN MEISTER VON BALK
 37 DATA1,12,2,12,1,13,2,13,1,14,2,14,2
,18,3,18,4,18,5,20,6,20,7,20,8,20
 40 DATA9,20,10,20,11,20,12,20,13,20,14
,18,15,18,16,18,18,16,19,16,20,16
 43 DATA21,16,22,16,23,16,37,3,37,4,37,
5
 45 FORI=1TONG:READG(I,2):READG(I,1):NE
XT:CR=1:CG=1:GOSUB3000
 47 IFCD=1THENGOSUB10900:GOTO155
 50 DATA36,"-/",33,"-◤ ■",31,"--◤
■",30,"--◤    ■",30,"-◤    ◤"
 53 DATA19,"◤  ◥ ---  --▒  ◤---",16
,"◤     ◥  ----- ▒  ◥---"
 55 DATA13,"◤                ◥---  ◤ ■---
-",9,"◤                ■-- ▒  ◤--"
 57 DATA8,"◤                        ◤----
-",1,"—▒                       ◤------
-"
 60 DATA3,"▒                         ■----
---",3,"▒                      ◤-- - -"
 63 DATA3,"▒                 ◤  ◥▒  ◤ --"
,3,"▒          ◤"
 65 DATA3,"◥▒               ■",4,"◥▒
  ◤",5,"◥▒          ◤",6,"◥▒     ◤"
 100 PRINT"▒":PRINT:FORI=0TO18:READT:RE
ADC$:PRINTW$(I+2)TAB(T)C$:NEXT
 110 FORI=1TO37:FORJK=0TO1:POKE32728+I+
JK*920+TP(JK)*40,192:NEXT:NEXT
```

Seguito "Battaglia del lago ghiacciato".

```
 120 POKE32768,240:POKE32806,238:POKE33
608,237:POKE33646,253:FORI=1TO20
 130 PRINTW$(I+1)"▒|■"TAB(38)"▒|■":NEXT
:FORX1=1TO37:FORY1=1TO20
 140 C(Y1,X1,1)=PEEK(32768+Y1*40+X1):IF
C(Y1,X1,1)<>160THENC(Y1,X1,2)=99:GOTO14
7
 145 C(Y1,X1,2)=GH+RND(RND(TI)+RND(X1)+
RND(Y1))*VH
 147 IFC(Y1,X1,1)=45THENC(Y1,X1,2)=90
 150 NEXT:NEXT
 155 FORI=1TONR:C1=RN:IFR(I,0)<-3THEN15
9
 156 IFPEEK(32768+R(I,1)*40+R(I,2))=160
THENC1=RR
 157 POKE32768+R(I,1)*40+R(I,2),C1
 159 NEXT
 160 FORI=1TONG:C1=GN:IFG(I,0)<-3THEN16
9
 163 IFPEEK(32768+G(I,1)*40+G(I,2))=160
THENC1=GR
 165 POKE32768+G(I,1)*40+G(I,2),C1
 169 NEXT:IFMC$="G"THEN195
 175 IFR(CR,0)<=0THEN190
 177 IFR(CR,4)>0THENR(CR,4)=R(CR,4)-1:G
OTO190
 180 C$="R":GOSUB500:GOSUB38000
 190 CR=CR+1:IFCR>NRTHENCR=1
 195 IFG(CG,0)<=0THEN210
 197 IFG(CG,4)>0THENG(CG,4)=G(CG,4)-1:G
OTO210
 200 C$="G":GOSUB500:GOSUB38000
 210 CG=CG+1:IFCG>NGTHENCG=1
 220 GOSUB4000:GOTO175
 500 REM GESTIONE UNITA'
 510 IFC$="G"THENNC=GN:RC=GR:X=G(CG,2):
Y=G(CG,1):P=G(CG,3)
 515 IFC$="R"THENNC=RN:RC=RR:X=R(CR,2):
Y=R(CR,1):P=R(CR,3)
 520 X0=X:Y0=Y:GOSUB5000:IFC$=JK$THENGO
SUB6000:GOTO540
 530 GOSUB1000
 540 IFC$="R"THENR(CR,2)=X:R(CR,1)=Y
 550 IFC$="G"THENG(CG,2)=X:G(CG,1)=Y
 560 GOSUB8000
 999 RETURN
 1000 REM INPUT
 1010 FG=0:PRINTW$(24)"DIREZIONE (1-9)/
 FINE (0)"
 1020 GOSUB5000:GETDR$:IFDR$=""THEN1020
 1030 DR=VAL(DR$):IFDR<0ANDDR>9THEN1020
 1035 IFDR=0THENGOTO11000
 1040 GOSUB9000:GOSUB30000
 1050 GOSUB17000:IFFG=1THENRETURN
```



La battaglia del lago ghiacciato

Seguito "Battaglia del lago ghiacciato".

```
 1055 GOSUB7000:IFFG=1THENRETURN
 1060 GOSUB15000:TC=RC:RC=NC:NC=TC:GOSU
B5000
 1999 RETURN
 3000 FORI=1TONR:R(I,3)=1:NEXT:FORI=1TO
NG:G(I,3)=8:NEXT:RR$="ALEXANDER NEVSKIJ
"
 3003 FORI=1TONR:R(I,0)=3:NEXT:FORI=1TO
NG:G(I,0)=15:NEXT:PR=NR:PG=NG
 3005 BL$="▣LA BATTAGLIA DEL LAGO GHIAC
CIATO■":G$="▣GROSSEN MEISTER VON BALK■"
 3007 R0$="PRINCIPE ALEXANDER NEVSKIJ"
 3010 PRINT"▣"TAB(4)BL$:PRINTW$(3)TAB(8
)"DALL'"CHR$(34)RR$CHR$(34)
 3020 W=11:PRINTW$(5)TAB(11)"DI S.M. EJ
ZENSTEJN"
 3030 PRINTW$(7)TAB(3)"HAI BATTAGLIE IN
 CORSO";:INPUTR$
 3040 R$=LEFT$(R$,1):IFR$<>"S"ANDR$<>"N
"THEN3030
 3050 IFR$="S"THENGOSUB10500:RETURN
 3100 PRINTW$(W)TAB(3)"VUOI INTERPRETAR
E LA PARTE DI:"
 3110 PRINTW$(W+2)TAB(3)"▣R■ -"R0$:PRIN
TW$(W+4)TAB(3)"▣G■ -"G$
 3120 GETR$:IFR$=""THEN3120
 3130 IFR$<>"G"ANDR$<>"R"THEN3120
 3140 JG$="R":JK$="G":IFR$="G"THENJG$="
G":JK$="R"
 3150 RETURN
 4000 REM CONDIZIONE DI TERMINAZIONE
 4005 IFCG<>1THENRETURN
 4010 GOSUB35000:IFJK$="G"THENS$=STR$(R
7)+"-▣"+STR$(G7)+"■"
 4020 IFJK$="R"THENS$="▣"+STR$(G7)+"■-"
+STR$(R7)
 4030 PRINTW$(24)"PUNTEGGIO :"S$:GOSUB1
3000:GOSUB9000
 4040 IFGP>RP*3THEN36000
 4050 IFRP>=GPTHENGOSUB37000
 4999 RETURN
 5000 REM DISPLAY
 5010 FORQB=1TO30:POKE32768+Y*40+X,NC:P
OKE32768+Y*40+X,RC:NEXT:RETURN
 6000 FG=0:XK=X:YK=Y:FORDR=1TO9:X=XK:Y=
YK:GOSUB30000:DD(DR)=0
 6016 IFC$="R"THENDD(DR)=DD(DR)+(Y-YK)*
75
 6017 IFC$="G"THENDD(DR)=DD(DR)+(YK-Y)*
75
 6019 K9=RND(TI+P1+P2+X+Y):DD(DR)=DD(DR
)-ABS(X-XK)*10-ABS(Y-10)*50-ABS(X-10)*5
 6020 EE=PEEK(32768+Y*40+X):IFEE=160THE
NDD(DR)=DD(DR)-1
 6025 IFEE=160ANDDR=5ANDC$="G"THENDD(DR
)=-9999:GOTO6050
 6030 IFX<10ORX>37ORY<10ORY>20THENDD(DR)=
-999
 6035 IFEE=214THENDD(DR)=-9999:GOTO6050
 6037 IFEE=45ANDC$="G"THENDD(DR)=-9999:
GOTO6050
```

Seguito "Battaglia del lago ghiacciato".

```
 6040 IF(EE=81OREE=87OREE=215OREE=209)A
NDDR<>5THENDD(DR)=-9999:GOTO6050
 6046 PP=0:GOSUB31000:IFNE=0THENDD(DR)=
DD(DR)+AM/25+K9-.5-PP:GOTO6050
 6047 GOSUB32000:DD(DR)=(P1-P2)*100+K9-
.5-PP
 6050 NEXT:PM=-10000:DR=INT(RND(TI)*9+1
):FORRD=1TO9
 6060 IFDD(RD)>=PMTHENPM=DD(RD):DR=RD
 6070 NEXT:X=XK:Y=YK:GOTO1040
 7000 REM GHIACCIO SI ROMPE ?
 7005 FL=0:IFC(Y0,X0,2)>=90THEN7020
 7010 C(Y0,X0,2)=C(Y0,X0,2)-P/2:X1=X0:Y
1=Y0:REM  GOSUB2000
 7020 IFC(Y,X,2)=99THENRETURN
 7030 IFC(Y,X,2)=90ANDC$="G"THENGOSUB14
000:RETURN
  040 TC=NC:NC=RC:RC=TC:IFP>=C(Y,X,2)TH
ENGOSUB12000
 7999 RETURN
 8000 IFFL=1THENFL=0:RETURN
 8010 GOSUB31000
 8110 IFNE=0THENRETURN
 8120 GOSUB32000
 8130 K1=P2*100/(P1+P2):K2=P1*100/(P1+P
2):K=RND(TI+P1+P2)*100
 8140 IFK<=K1/3THENGOSUB22000:RETURN
 8150 IFK>K1/3ANDK<=K1*.66THENGOSUB2300
0:RETURN
 8154 IFK>K1*.66ANDK<=K1THENGOSUB39000:
RETURN
 8156 IFK>K1ANDK<=(K1+K2/3)THENGOSUB400
00:RETURN
 8160 IFK>(K1+K2/3)ANDK<=(K1+K2*.66)THE
NGOSUB24000:RETURN
 8170 IFK>(K1+K2*.66)THENGOSUB25010:RET
URN
 8500 AM(AM,1)=Y+DY:AM(AM,2)=X+DX:D$=C$
:GOSUB20000:AM(AM,0)=W:AM(AM,3)=H
 8510 AM(AM,4)=K:RETURN
 8600 NE(NE,1)=Y+DY:NE(NE,2)=X+DX:D$=N$
:GOSUB20000:NE(NE,0)=W:NE(NE,3)=H
 8610 NE(NE,4)=K:RETURN
 8999 RETURN
 9000 FORQQ=33688TO33728:POKEQQ,32:NEXT
:RETURN
 10500 REM CARICAMENTO DA PROGRAMMA
 10510 CD=1:MC$=C$:PRINT"▣"W$(4)"NOME D
ISTINTIVO DEL FILE":INPUTN$
 10520 PRINTW$(7)"RIAVVOLGETE LA CASSET
TA CONTENENTE IL":PRINT"FILE DATI."
 10530 PRINT"PRONTI";:INPUTZ$:IFLEFT$(Z
$,1)<>"S"THEN10520
 10540 OPEN1,1,0,NF$:INPUT#1,CR:INPUT#1
,CG:INPUT#1,PG:INPUT#1,PR
 10550 INPUT#1,MC$:INPUT#1,JK$:INPUT#1,
JG$:FORI1=1TO20:FORI2=1TO37:FORI3=1TO2
 10560 INPUT#1,C(I1,I2,I3):NEXT:NEXT:NE
XT
 10565 INPUT#1,R6:INPUT#1,G6
 10570 FORI1=1TONR:FORI2=0TO4:INPUT#1,R
```



La battaglia del lago ghiacciato

Seguito "Battaglia del lago ghiacciato".

```
(I1,I2):NEXT:NEXT
 10580 FORI1=1TONG:FORI2=0TO4:INPUT#1,G
(I1,I2):NEXT:NEXT:CLOSE1
 10590 RETURN
 10900 PRINT"▣":FORI=1TO37:FORJK=0TO1:P
OKE32728+I+JK*920+TP(JK)*40,192:NEXT:NE
XT
 10920 POKE32768,240:POKE32806,238:POKE
33608,237:POKE33646,253:FORI=1TO20
 10930 PRINTW$(I+1)"▨|■"TAB(38)"▨|■":NE
XT:FORX1=1TO37:FORY1=1TO20
 10940 POKE(32768+Y1*40+X1),C(Y1,X1,1):
NEXT:NEXT:RETURN
 11000 REM SALVATAGGIO SU NASTRO
 11010 MC$=C$:PRINT"▣"W$(4)"NOME DISTIN
TIVO DEL FILE":INPUTN$
 11020 PRINTW$(7)"PROCURATEVI UNA CASSE
TTA E RIAVVOLGETELA"
 11030 PRINT"PRONTI";:INPUTZ$:IFLEFT$(Z
$,1)<>"S"THEN11020
 11040 OPEN1,1,2,NF$:PRINT#1,CR:PRINT#1
,CG:PRINT#1,PG:PRINT#1,PR
 11050 PRINT#1,MC$:PRINT#1,JK$:PRINT#1,
JG$:FORI1=1TO20:FORI2=1TO37:FORI3=1TO2
 11060 PRINT#1,C(I1,I2,I3):NEXT:NEXT:NE
XT
 11065 PRINT#1,R6:PRINT#1,G6
 11070 FORI1=1TONR:FORI2=0TO4:PRINT#1,R
(I1,I2):NEXT:NEXT
 11080 FORI1=1TONG:FORI2=0TO4:PRINT#1,G
(I1,I2):NEXT:NEXT:CLOSE1
 11090 PRINTW$(15)"VUOI CONTINUARE";:IN
PUTZ$:Z$=LEFT$(Z$,1)
 11100 IFZ$<>"S"ANDZ$<>"N"THEN11090
 11110 IFZ$="S"THENCD=1:GOTO47
 11999 END
 12000 C(Y,X,2)=0:NC=RC:RC=214:POKE3276
8+Y0*40+X0,C(Y0,X0,1):GOSUB5000
 12005 C(Y,X,1)=214
 12010 PRINTW$(24)"IL GHIACCIO SI SPEZZ
A!":GOSUB13000:GOSUB9000:FG=1
 12020 Q=-4:GOSUB16000:FL=1:RETURN
 13000 FORY9=0TO1000:NEXT:RETURN
 14000 C(Y,X,1)=NC:POKE32768+Y0*40+X0,C
(Y0,X0,1):TC=NC:NC=RC:RC=TC:GOSUB5000
 14010 PRINTW$(24)"PALUDE!":GOSUB13000:
GOSUB9000:FG=1
 14020 Q=-3:GOSUB16000:RETURN
 15000 POKE32768+Y0*40+X0,C(Y0,X0,1):RE
TURN
 16000 IFC$="R"THENR(CR,0)=0:PR=PR-1:R6
=R6-1:RETURN
 16010 IFC$="G"THENG(CG,0)=0:PG=PG-1:G6
=G6-1:RETURN
 16999 RETURN
 17000 IFX<1ORX>37ORY<1ORY>20THENGOSUB1
8000
 17005 PK=PEEK(32768+Y*40+X):IFPK=81ORP
K=87ORPK=215ORPK=209THENX=X0:Y=Y0
 17010 RETURN
 18000 POKE32768+Y0*40+X0,C(Y0,X0,1)
 18010 PRINTW$(24)"L'UNITA' E' FUGGITA"
```

Seguito "Battaglia del lago ghiacciato".

```
:GOSUB13000:GOSUB9000:FG=1
 18020 Q=-5:GOSUB16000:RETURN
 19000 IFNA=81ORNA=215THENNN=87:R9=209
 19010 IFNA=87ORNA=209THENNN=81:R9=215
 19020 RETURN
 20000 W=1:IFD$="G"THEN20500
 20020 IFW>NRTHENRETURN
 20030 IF(R(W,1)=Y+DY)AND(R(W,2)=X+DX)A
ND(R(W,0)>0)THENH=R(W,3):K=R(W,0):RETUR
N
 20040 W=W+1:GOTO20020
 20500 IFW>NGTHENRETURN
 20530 IF(G(W,1)=Y+DY)AND(G(W,2)=X+DX)A
ND(G(W,0)>0)THENH=G(W,3):K=G(W,0):RETUR
N
 20540 W=W+1:GOTO20500
 21000 IFC$="R"THENN$="G"
 21010 IFC$="G"THENN$="R"
 21020 RETURN
 22000 REM MORTE AMICI
 22010 FORZ=1TOAM:GOSUB28000:NEXT:RETUR
N
 23000 REM FERIMENTO O DISARCIONAMENTO
AMICI
 23010 IFC$="R"THEN23500
 23020 FORZ=1TOAM:IN=AM(Z,0):IFG(IN,3)=
3THENGOSUB33500:GOTO23040
 23025 GOSUB26000:RC=PEEK(32768+AM(Z,1)
*40+AM(Z,2)):NC=C(AM(Z,1),AM(Z,2),1)
 23027 G(IN,0)=G(IN,0)-12:XX=AM(Z,2):YY
=AM(Z,1):GOSUB27000:G(IN,4)=1
 23030 FORTY=1TO2:PRINTW$(24)TY$(TY):GO
SUB13000:GOSUB9000:NEXT
 23040 NEXT:RETURN
 23500 FORZ=1TOAM:IN=AM(Z,0):IFR(IN,0)=
1.5THENGOSUB28000:GOTO23540
 23510 R(IN,0)=R(IN,0)-1.5:RC=PEEK(3276
8+AM(Z,1)*40+AM(Z,2)):R(IN,4)=2
 23520 NC=C(AM(Z,1),AM(Z,2),1)
 23530 XX=AM(Z,2):YY=AM(Z,1):GOSUB27000
 23535 PRINTW$(24);:FORTY=1TO2:PRINTYT$
(TY)" ";:NEXT:GOSUB13000:PRINT:GOSUB900
0
 23540 NEXT:RETURN
 24000 REM RITIRATA O DISARCIONAMENTO N
EMICI
 24010 IFN$="R"THEN24500
 24020 FORZ=1TONE:IN=NE(Z,0):IFG(IN,3)=
3THENGOSUB34500:GOTO24040
 24025 GOSUB26000:RC=PEEK(32768+NE(Z,1)
*40+NE(Z,2)):NC=C(NE(Z,1),NE(Z,2),1)
 24027 G(IN,0)=G(IN,0)-12:XX=NE(Z,2):YY
=NE(Z,1):GOSUB27000:G(IN,4)=1
 24030 FORTY=1TO2:PRINTW$(24)TY$(TY):GO
SUB13000:GOSUB9000:NEXT
 24040 NEXT:RETURN
 24500 FORZ=1TONE:IN=NE(Z,0):IFR(IN,0)=
1.5THENGOSUB29000:GOTO24540
 24510 R(IN,0)=R(IN,0)-1.5:RC=PEEK(3276
8+NE(Z,1)*40+NE(Z,2)):R(IN,4)=2
 24520 NC=C(NE(Z,1),NE(Z,2),1)
 24530 XX=NE(Z,2):YY=NE(Z,1):GOSUB27000
```



La battaglia del lago ghiacciato

Seguito "Battaglia del lago ghiacciato".

```
24535 PRINTW$(24);:FORTY=1TO2:PRINTYT$(TY)" ";:NEXT:GOSUB13000:PRINT:GOSUB9000
24540 NEXT:RETURN
25000 IFD$="R"THENR(IN,0)=W:PR=PR-1:R6=R6-1:RETURN
25005 IFD$="G"THENG(IN,0)=W:PG=PG-1:G6=G6-1:RETURN
25010 FORZ=1TONE:GOSUB29000:NEXT:RETURN
26000 G(IN,3)=3:RETURN
27000 FOROB=1TO30:POKE32768+YY*40+XX,NC:POKE32768+YY*40+XX,RC:NEXT:RETURN
28000 NC=PEEK(32768+AM(Z,1)*40+AM(Z,2)):RC=C(AM(Z,1),AM(Z,2),1)
28010 YY=AM(Z,1):XX=AM(Z,2):GOSUB27000
28020 D$=C$:IN=AM(Z,0):W=-5:GOSUB25000:RETURN
29000 NC=PEEK(32768+NE(Z,1)*40+NE(Z,2)):RC=C(NE(Z,1),NE(Z,2),1)
29010 YY=NE(Z,1):XX=NE(Z,2):GOSUB27000
29020 D$=N$:IN=NE(Z,0):W=-5:GOSUB25000:RETURN
```

Seguito "Battaglia del lago ghiacciato".

```
30000 ONDRGOTO30010,30020,30030,30040,30050,30060,30070,30080,30090
30010 Y=Y+1:X=X-1:RETURN
30020 Y=Y+1:RETURN
30030 Y=Y+1:X=X+1:RETURN
30040 X=X-1:RETURN
30050 RETURN
30060 X=X+1:RETURN
30070 Y=Y-1:X=X-1:RETURN
30080 Y=Y-1:RETURN
30090 Y=Y-1:X=X+1:RETURN
31000 GOSUB21000
31010 AM=0:NE=0:NA=NC:RA=RC:GOSUB19000:FORDY=-1TO1:IFY+DY<1ORY+DY>20THEN31100
31015 FORDX=-1TO1:IFX+DX<1ORX+DX>37THEN31090
31020 JK=PEEK(32768+(Y+DY)*40+(X+DX)):IFJK=NAORJK=RATHENAM=AM+1:GOSUB8500
31030 IFJK=NNORJK=R9THENNE=NE+1:GOSUB8600
31040 IFJK=45THENPP=PP+100
31090 NEXTDX
31100 NEXTDY
31110 RETURN
32000 P1=0:P2=0:IFAM=0THEN32015
32010 FORTY=1TOAM:P1=P1+AM(TY,4):NEXT
32015 IFNE=0THEN32999
32020 FORTY=1TONE:P2=P2+NE(TY,4):NEXT
32999 RETURN
33500 IFG(IN,0)=1.5ORG(IN,0)=13.5THENGOSUB28000:GOTO33540
33510 G(IN,0)=G(IN,0)-1.5:RC=PEEK(32768+AM(Z,1)*40+AM(Z,2)):G(IN,4)=2
33520 NC=C(AM(Z,1),AM(Z,2),1)
33530 XX=AM(Z,2):YY=AM(Z,1):GOSUB27000
33535 PRINTW$(24)TY$(1):GOSUB13000:GOSUB9000:PRINTW$(24)YT$(2)
33537 GOSUB13000:GOSUB9000
33540 RETURN
34500 IFG(IN,0)=1.5ORG(IN,0)=13.5THENGOSUB28000:GOTO34540
34510 G(IN,0)=G(IN,0)-1.5:RC=PEEK(32768+NE(Z,1)*40+NE(Z,2)):G(IN,4)=2
34520 NC=C(NE(Z,1),NE(Z,2),1)
34530 XX=NE(Z,2):YY=NE(Z,1):GOSUB27000
34535 PRINTW$(24)TY$(1):GOSUB13000:GOSUB9000:PRINTW$(24)YT$(2)
34537 GOSUB13000:GOSUB9000
34540 RETURN
35000 RP=0:FORI=1TONR:RX=R(I,0):IFRX<0THENRX=0
35010 RP=RP+RX:NEXT:GP=0:FORI=1TONG:GX=G(I,0):IFGX<0THENGX=0
35020 GP=GP+GX:NEXT:GP=GP+PG:RP=RP+PR:R7=MG-GP:G7=MR-RP:RETURN
36000 PRINTW$(24)"▒VITTORIA DELLA CAVALLERIA TEUTONICA■":GOSUB13000:GOSUB9000
36010 PRINTW$(24)"▒DEL ■"G$
36999 FORHJ=1TO50000:NEXT:END
37000 PRINTW$(24)"VITTORIA DELLA FANTERIA RUSSA":GOSUB13000:GOSUB9000
37010 PRINTW$(24)"DEL "R0$
```

Seguito "Battaglia del lago ghiacciato".

```
 37999 FORHJ=1TO50000:NEXT:END
 38000 IFR6=0THENGOSUB36000
 38010 IFG6=0THENGOSUB37000
 38020 GOSUB41000:RETURN
 39000 REM FERIMENTO AMICI
 39010 IFC$="R"THENGOSUB23500:RETURN
 39020 FORZ=1TOAM:IN=AM(Z,0):GOSUB33500
:NEXT:RETURN
 40000 REM FERIMENTO NEMICI
 40010 IFN$="R"THENGOSUB24500:RETURN
 40020 FORZ=1TONE:IN=NE(Z,0):GOSUB34500
:NEXT:RETURN
 41000 IFC$="R"THEN41500
 41010 IFG(CG,3)>3THENRETURN
 41020 IFG(CG,0)>1.5THENG(CG,4)=1:RETUR
N
 41030 G(CG,4)=2:RETURN
 41500 IFR(CR,0)>1.5THENRETURN
 41510 R(CR,4)=2:RETURN
 59000 REM DA ELIMINARE IN RIGA 6050
 59005 DD=INT(DD(DR)*1000)/1000
 59006 K9=INT(K9*1000)/1000
 59010 PRINTW$(24)DRTAB(3)DDTAB(10)AMTA
B(15)NETAB(20)P1TAB(25)P2TAB(30)K9
 59020 GETTW$:IFTW$=""THEN59020
 59999 GOSUB9000:P1=0:P2=0:RETURN
```

Lista simboli grafici

```
     0 : 1  HOME =CHR$(19)
         24  CRSR↑ =CHR$(17)

    50 : 1  REVERSE =CHR$(18)
         1  SHIFT REVERSE =CHR$(146)
         1  REVERSE =CHR$(18)
         1  SHIFT REVERSE =CHR$(146)
         1  REVERSE =CHR$(18)
         1  SHIFT REVERSE =CHR$(146)
         1  REVERSE =CHR$(18)
         1  SHIFT REVERSE =CHR$(146)

    53 : 1  REVERSE =CHR$(18)
         1  SHIFT REVERSE =CHR$(146)
         1  REVERSE =CHR$(18)
         1  SHIFT REVERSE =CHR$(146)
         1  REVERSE =CHR$(18)
         1  SHIFT REVERSE =CHR$(146)
         1  REVERSE =CHR$(18)
         1  SHIFT REVERSE =CHR$(146)

    55 : 1  REVERSE =CHR$(18)
         1  SHIFT REVERSE =CHR$(146)
         1  REVERSE =CHR$(18)
         1  SHIFT REVERSE =CHR$(146)
         1  REVERSE =CHR$(18)
         1  SHIFT REVERSE =CHR$(146)
         1  REVERSE =CHR$(18)
         1  SHIFT REVERSE =CHR$(146)

    57 : 1  REVERSE =CHR$(18)
         1  SHIFT REVERSE =CHR$(146)
         1  REVERSE =CHR$(18)
         1  SHIFT REVERSE =CHR$(146)
```

Seguito "Battaglia del lago ghiacciato".

```
    60 : 1  REVERSE =CHR$(18)
         1  SHIFT REVERSE =CHR$(146)
         1  REVERSE =CHR$(18)
         1  SHIFT REVERSE =CHR$(146)

    63 : 1  REVERSE =CHR$(18)
         1  SHIFT REVERSE =CHR$(146)
         1  REVERSE =CHR$(18)
         1  SHIFT REVERSE =CHR$(146)
         1  REVERSE =CHR$(18)
         1  SHIFT REVERSE =CHR$(146)

    65 : 1  REVERSE =CHR$(18)
         1  SHIFT REVERSE =CHR$(146)
         1  REVERSE =CHR$(18)
         1  SHIFT REVERSE =CHR$(146)
         1  REVERSE =CHR$(18)
         1  SHIFT REVERSE =CHR$(146)
         1  REVERSE =CHR$(18)
         1  SHIFT REVERSE =CHR$(146)

   100 : 1  SHIFT HOME =CHR$(147)

   130 : 1  REVERSE =CHR$(18)
         1  SHIFT REVERSE =CHR$(146)
         1  REVERSE =CHR$(18)
         1  SHIFT REVERSE =CHR$(146)

  3005 : 1  REVERSE =CHR$(18)
         1  SHIFT REVERSE =CHR$(146)
         1  REVERSE =CHR$(18)
         1  SHIFT REVERSE =CHR$(146)

  3010 : 1  SHIFT HOME =CHR$(147)

  3110 : 1  REVERSE =CHR$(18)
         1  SHIFT REVERSE =CHR$(146)
         1  REVERSE =CHR$(18)
         1  SHIFT REVERSE =CHR$(146)

  4010 : 1  REVERSE =CHR$(18)
         1  SHIFT REVERSE =CHR$(146)

  4020 : 1  REVERSE =CHR$(18)
         1  SHIFT REVERSE =CHR$(146)

 10510 : 1  SHIFT HOME =CHR$(147)

 10900 : 1  SHIFT HOME =CHR$(147)

 10930 : 1  REVERSE =CHR$(18)
            SHIFT REVERSE =CHR$(146)
            REVERSE =CHR$(18)
            SHIFT REVERSE =CHR$(146)

 11010 : 1  SHIFT HOME =CHR$(147)

 36000 : 1  REVERSE =CHR$(18)
         1  SHIFT REVERSE =CHR$(146)

 36010 : 1  REVERSE =CHR$(18)
         1  SHIFT REVERSE =CHR$(146)
```

PERSONAL SOFTWARE

ZX81 Croupier

Giocate alla roulette con il vostro personal Sinclair

di *Angelo Motta*

Per giocare d'azzardo senza correre il rischio di essere imbrogliati si può chiedere allo ZX81 di tenere banco per una serata. È sufficiente introdurre il programma presentato, che ripropone fedelmente il gioco della roulette con tutti i possibili tipi di scommesse, tenendo conto anche, in caso esca il numero zero, delle combinazioni semplici che rimangono sul tappeto per il gioco successivo. Le regole sono talmente semplici ed il gioco talmente noto che è inutile dilungarsi ora in spiegazioni comunque contenute nel programma. La figura 1 illustra i tipi di scommesse possibili con le relative vincite.

Alla roulette possono partecipare fino a 10 giocatori che possono effettuare una sola scommessa per turno. È possibile aumentare il numero dei partecipanti variando i dimensionamenti delle matrici effettuati dalla linea 700 in poi.

Oltre al gioco in se stesso, il programma presenta due routine in linguaggio macchina, contenute nella REM iniziale, che consentono una gestione del video più veloce e inoltre possono essere utilizzate anche in altri programmi.

La prima salva il contenuto del Display File (o di parte di esso) nella parte alta della RAM e lo richiama al bisogno, aumentando la velocità con cui si può cambiare immagine; la seconda consente il lampeggio del video mediante inversione.

Descrizione	Importo della vincita
EN PLEIN (= Pieno) Pieno su un numero, cioè puntata su di un solo numero.	35 volte la posta
CHEVAL (= Cavallo) 2 numeri collegati (Es. 1 e 2)	17 volte la posta
TRASVERSALE PLEINE (= Piena trasversale o Terzina) Fila orizzontale di 3 numeri (Es. 1 - 2 - 3 o 19 - 20 - 21 - ecc.).	11 volte la posta
CARRÉ (= Quadrato) Blocco di 4 numeri in quadrato (Es. 1 - 2 - 4 - 5 o 2 - 3 - 5 - 6 - ecc.).	8 volte la posta
TRASVERSALE SIMPLE (= Trasversale semplice o testina) 2 file orizzontali di 3 numeri, una sotto l'altra (Es. 1 - 2 - 3 - 4 - 5 - 6).	5 volte la posta
DOUZAINE (= Dozzina) 12P = Numeri 1/12 (prima dozzina) 12M = Numeri 13/24 (seconda dozzina) 12D = Numeri 25/36 (terza dozzina).	2 volte la posta
COLONNE 12 numeri in verticale: Prima colonna : 34 Seconda colonna : 35 Terza colonna : 36.	2 volte la posta
Combinazioni semplici (vengono tutte pagate 1 volta la posta)	
PAIR = Numeri pari **IMPAIR** = Numeri dispari **MANQUE** = Numeri dall'1 al 18 (metà superiore del banco) **PASSE** = Numeri dal 19 al 36 (metà inferiore del banco) **ROUGE** = Rosso **NOIR** = Nero	

Se esce lo 0, le puntate relative alle combinazioni semplici rimangono sul tavolo e poi, se col gioco successivo escono quelle combinazioni, esse vengono pagate al giocatore, altrimenti vanno al banco.

Per le puntate 2, 3, 4, 5, il numero giocato indica l'angolo basso a sinistra della figura giocata (per il quadrato 1 - 2 - 4 - 5 occorre giocare il numero 1).

Figura 1. *Istruzioni di gioco.*

Analizziamo la prima routine illustrata nella figura 2.

Si basa su una delle più potenti istruzioni presenti sullo Z80 per il trasferimento di dati e, precisamente, l'istruzione LDIR. Ecco qualche spiegazione per i neofiti del linguaggio macchina.

Questa istruzione consente il trasferimento rapido di un blocco di dati da una zona di memoria ad un'altra. Per la sua esecuzione vengono utilizzati i tre registri doppi BC, DE e HL. Il registro HL punta alla prima locazione di memoria contenente il blocco dati da trasferire; il registro DE

alla prima locazione di memoria dove saranno trasferiti quei dati; il registro BC il numero di byte da trasferire. Dopo ogni trasferimento, i registri DE e HL sono incrementati di "1" ed il registro BC decrementato fino a quando BC non contiene 0.

Se analizziamo la nostra routine vediamo che essa si compone di due parti. La prima trasferisce il Dispaly File in altra zona di memoria e pertanto il registro HL deve puntare all'inizio del Display File ed il registro DE alla zona di RAM designata. La seconda parte della routine riporta i dati trasferiti in RAM nel Display File, e deve quindi avere i valori dei registri DE & HL invertiti. In entrambi i casi il registro BC (contatore) conterrà la lunghezza del Display File.

La routine è stata inserita nel programma nella REM iniziale fra le locazioni da 16550 a 16570, ma può essere inserita in un qualsiasi punto di una REM o, addirittura in una stringa, basta saperla chiamare nel modo giusto.

Le locazioni n. 8 e n. 9 della routine contengono il valore del registro BC, cioè il numero dei byte da trasferire.

Se eseguiamo:

PRINT PEEK 16557 + 256 ★ PEEK 16558 (NEW LINE)

otterremo il valore 594. Siccome ogni linea del Display File è composta da 33 caratteri (32 + New Line), dividendo il valore ottenuto per 33 (593/33) si ottiene il numero delle linee che nel programma Roulette viene salvato in RAM, cioè 18.

Variando il valore dei registri BC si può utilizzare la routine per salvare un numero a piacere di linee del Display File.

Facciamo un esempio: se volessimo trasferire tutte le 24 linee dello schermo, dovremmo ragionare così:

24 x 33 = 792 (= numero dei byte che compongono lo schermo)
INT (792/256) = 3 (MSB o numero da inserire nel registro B)
792 − 256 ★ 3 = 24 (LSB o numero da inserire nel registro C)

I valori ottenuti devono essere inseriti con l'istruzione POKE nelle locazioni n. 8 (per il registro C) e 9 (per il registro B) della routine, ottenen-

Routine di trasferimento del Display File nella RAM alta.

16550	42	12	64	2A	0C	40	LD HL, (16396)
16553	17	0	122	11	00	7A	LD DE, 31232
16556	1	82	2	01	52	02	LD BC, 594
16559	237	176		ED	B0		LDIR
16561	201			C9			RET

Routine di trasferimento dalla RAM alta al Display File.

16562	237	91	12	42/ED	5B	0C 40	LD DE, (16396)
16566	33	0	122	21	00	7A	LD HL, 31232
16569	24	241		18	F1		JR − 15

Per aumentare o diminuire le linee del Display File da trasferire variare il dato dei registri BC (contatore). Se per esempio volessimo trasferire tutte e 22 le linee dovremo inserire nella locazione 16557 il valore 214 decimale e nella locazione 16557 il valore 2 decimale. 214 + 2 ★ 256 = 726 = il numero dei caratteri contenuti nelle prime 22 righe del Display File.

Figura 2. *Routine in linguaggio macchina per il trasferimento del Display File in RAM e viceversa.*

16571	42	12	64	2A	0C	40	LD HL, (16396)
16574	6	18		06	12		LD B, 18
16576	197			C5			PUSH BC
16577	6	32		06	20		LD B, 32
16579	35			23			INC HL
16580	126			7E			LD A, (HL)
16581	254	127		FE	7F		CP 127
16583	56	4		38	04		JR C + 4
16585	222	128		DE	80		SBC A, 128
16587	24	2		18	02		JR + 2
16589	198	128		C6	80		ADD A, 128
16591	119			77			LD (HL), A
16592	16	241		10	F1		DJNZ − 15
16594	35			23			INC HL
16595	193			C1			POP BC
16596	16	234		10	EA		DJNZ − 22
16598	201			C9			RET

Nella locazione 16575 è contenuto il numero delle linee del video che si vogliono invertire (nel nostro caso 18).
Per aumentare o diminuire le linee da invertire, variare tale valore con l'istruzione:
POKE 16575, 24 (tutte le 24 linee per esempio).
Inserendo la routine in un ciclo FOR ... NEXT (il ciclo dovrà sempre essere pari) si otterrà il lampeggio del video.

Figura 3. *Routine per l'inversione del video.*

Figura 4.

```
   1 REM RNRNRNRNRNNRNRNRNRRNRNR
NRNRNNRNRNRNRE£RND) ?▒?▒ GOSUB ▒
TAN GOSUB ?£RND5 ?/ LET E£RND▒>V
AL ▒47▒/ ▒LEN ▒?( LET 7AT ( REM T
AN 7)"▒ ▒▒▒▒▒▒▒
 100 REM
 101 REM  LANCIO DELLA REM
 102 REM INIZIALE
 103 REM
 110 FAST
 120 LET B$="2A0C4011007A015202E
D80C9ED580C4021007A18F12A0C40051
2C50620237EFE7F38040E801802C6807
710F123C110EAC92311080805020201 0
10101010 1"
 130 LET A=16550
 140 FOR I=1 TO LEN B$ STEP 2
 150 LET B=(CODE B$(I)-28)*16+CO
DE B$(I+1)-28
 160 POKE A,B
 170 LET A=A+1
 180 NEXT I
 200 REM
 201 REM VARIABILI INIZIALI
 202 REM
 210 POKE 16389,119
 220 CLS
 230 DIM M$(32)
 240 LET N$="▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒
▒▒▒▒▒▒"
 250 LET O$="▒        ▒        ▒
 ▒"
 260 LET DF=PEEK 16396+256*PEEK
16397
 300 REM
 301 REM LANCIO IN RAM TOP DELLO
 302 REM SCHERMO PUNTATE
 303 REM
 310 DIM C$(13,11)
 320 LET B$=" 1-EN PLEIN 2-CAVAL
LO  3-TERZINA  4-QUADRATO 5-SEST
INA  6-DOZZINA  7-COLONNA  8-PAR
I      9-DISPARI 10-MANQUE  11-PA
SSE   12-ROSSO   13-NERO"
 330 FOR I=1 TO 13
 340 LET C$(I)=B$(1+11*I-11 TO )
 350 NEXT I
 360 PRINT "PUNTATE E VINCITE(PO
STA X VOLTE)",,
 370 FOR I=1 TO 13
 380 PRINT C$(I);TAB 30;(" " AND
 PEEK (16598+I)<10);PEEK (16598+
I)
 390 NEXT I
 400 LET L=USR 16550
 410 CLS
 500 REM
 501 REM LANCIO IN RAM TOP DEL
 502 REM TAPPETO DI GIOCO
 503 REM
 510 PRINT TAB 3;N$;TAB 3;O$;" ▒
▒D▒"
 520 PRINT TAB 3;"▒MANQUE▒IMPAIR
▒▒▒▒▒▒ ▒M▒ ▒"
 530 PRINT TAB 3;O$;"P▒ ▒ ▒";TAB
 3;N$
 540 PRINT TAB 3;"▒▒6▒▒▒1▒▒▒
▒2▒▒▒3▒▒C  ▒";
 550 PRINT TAB 3;"▒▒ ▒▒▒5▒▒▒
▒4▒▒▒3▒▒       ";N$
 560 PRINT "▒0▒▒▒▒8▒1▒▒1▒2▒▒
2▒▒▒3▒B  ▒▒▒  ▒1▒▒7▒0▒▒
▒9▒▒▒5▒    ▒▒▒▒";N$
 570 PRINT TAB 3;"▒▒4▒▒▒1▒1▒▒▒
▒2▒▒2▒3▒▒▒A  ▒"
 580 PRINT TAB 3;"▒▒ ▒▒▒0▒3▒▒▒
▒2▒▒8▒1▒▒     ▒";TAB 3;N$
 590 PRINT TAB 3;O$;"P▒ ▒ ▒";TAB
 3;"▒ PASSE▒ PAIR ▒ NOIR ▒▒ ▒M▒
▒"
 600 PRINT TAB 3;O$;" ▒ ▒D▒";TAB
 3;N$
 610 POKE 16555,119
 620 LET L=USR 16550
 700 REM
 701 REM DIMENSIONAMENTO DELLE
 702 REM MATRICI
 703 REM
 710 DIM G$(10,10)
 720 DIM S(10)
 730 DIM P(10)
 740 DIM T(10)
 750 DIM S(10)
 760 FOR I=1 TO 10
 770 LET S(I)=1000
 780 NEXT I
 790 DIM N(10)
1000 REM
1001 REM INIZIO GIOCO
1002 REM
1010 CLS
1020 SLOW
1030 FOR I=1 TO 20
1040 PRINT AT 10,8;M$( TO 15)
1050 PRINT AT 10,8;"R O U L E T
T E"
1060 NEXT I
1070 PRINT ,,,TAB 2;"VOLETE LE I
STRUZIONI? (S/N)"
1080 GOTO 1080+(10 AND INKEY$="N
")+(6920 AND INKEY$="S")
1090 CLS
1100 PRINT "QUANTI GIOCATORI? MA
X 10)"
1110 INPUT NG
1120 IF NG<1 OR NG>10 THEN GOTO
1110
1130 PRINT ,,"NOME DEI GIOCATORI
:",,,
1140 FOR I=1 TO NG
1150 PRINT (" " AND I<10);I;" -
";
1160 INPUT G$(I)
1170 PRINT G$(I)
1180 NEXT I
1190 FOR I=1 TO 30
1200 NEXT I
1210 POKE 16566,119
1220 LET L=USR 16562
1230 FOR I=1 TO 30
1240 PRINT AT 19,0;M$
1250 PRINT AT 19,3;"[illegible]
[illegible]"
1260 NEXT I
1270 PRINT AT 19,0;M$
1500 REM
1501 REM INSERIMENTO PUNTATE
1502 REM
1510 FOR I=1 TO NG
1520 IF P(I)<>0 AND T(I)>7 THEN
GOTO 1750
1525 IF S(I)=0 THEN GOTO 1750
1530 PRINT AT 16,0;M$+M$+M$+M$
1550 PRINT AT 18,0;G$(I);" POSSI
EDI £ ";S(I)
1560 PRINT AT 21,0;"QUANTO PUNTI
?"
1570 INPUT PT
1580 IF PT>S(I) THEN GOTO 5000
1590 LET P(I)=PT
1595 IF P(I)=0 THEN GOTO 5200
1600 PRINT AT 19,0;"PUNTI £ ";PT
;AT 21,0;"SU QUALE TIPO DI PUNTA
TA?"
1610 POKE 16568,122
1620 LET L=USR 16562
1630 INPUT TP
1640 IF TP<1 OR TP>13 THEN GOTO
1000
1650 LET T(I)=TP
1660 PRINT AT 19,13;"SU ";C$(TP,
1 TO )
1670 PRINT AT 21,0;M$
1680 POKE 16566,119
1690 LET L=USR 16562
1710 IF T(I)<8 THEN GOSUB 6000+1
00*T(I)
1720 PRINT ;AT 21,0;"E' ESATTA L
A PUNTATA?";M$( TO 10)
1730 INPUT B$
1740 IF CODE B$=51 THEN GOTO 163
0
1750 NEXT I
1760 PRINT AT 16,0;M$+M$+M$+M$
3000 REM
3001 REM USCITA NUMERO
3010 RAND
3020 LET NU=INT (RND*37)
3030 LET DFA=DF+366+INT (NU/3.08
)*2+2-(99 AND NU-(INT (NU/3)*3)=
2)-(198 AND NU-(INT (NU/3)*3)=0)
3040 LET DFB=DFA+33
```

Figura 4.

```
3050 PRINT AT 19,7;"I GIOCHI SONO
 FATTI";AT 21,5;"NEW LINE PER LA
NCIARE"
3060 INPUT B$
3070 PRINT AT 19,0;M$;M$;M$
3080 FOR I=1 TO 120
3090 LET L=USR 16571
3100 NEXT I
3110 PRINT AT 19,0;"NUMERO USCIT
O:"
3115 IF NU=0 THEN GOTO 5100
3120 PRINT NU;" "+("ROSSO" AND P
EEK (16513+NU)=55)+("NERO" AND P
EEK (16513+NU)=51);
3130 PRINT (" PARI" AND NU=INT (
NU/2)*2)+(" DISPARI" AND NU<>INT
 (NU/2)*2);
3140 PRINT (" MANQUE" AND NU<=18
)+(" PASSE" AND NU>18)
3150 FOR I=1 TO 30
3160 POKE DFA,PEEK DFA+(128 AND
PEEK DFA<128)-(128 AND PEEK DFA>
128)
3170 POKE DFB,PEEK DFB+(128 AND
PEEK DFB<128)-(128 AND PEEK DFB>
=128)
3180 NEXT I
4000 REM
4001 REM AGGIORNAMENTO CAPITALI
4002 REM
4010 SCROLL
4020 SCROLL
4030 PRINT TAB 5;"AGGIORNAMENTO [illegible]
[illegible]"
4040 SCROLL
4050 FOR I=1 TO NG
4055 SCROLL
4060 LET V=0
4070 IF S(I)=0 THEN GOTO 4190
4080 IF P(I)=0 THEN GOTO 5900
4090 IF NU=0 AND T(I)>7 THEN GOT
O 5950
4100 GOSUB 5480+30*T(I)
4110 LET S(I)=S(I)-(P(I) AND V=0
)+(P(I)*PEEK (16598+T(I)) AND V=
1)
4120 LET T(I)=0
4130 LET P(I)=0
4140 SCROLL
4150 PRINT G$(I);(" VINCI" AND V
=1)+(" PERDI" AND V=0)
4160 SCROLL
4170 PRINT TAB 11;"ORA POSSIEDI
£. ";S(I)
4190 NEXT I
4200 SCROLL
4210 SCROLL
4220 PRINT "PREMETE NEWLINE PER
[illegible]MANCHE"
4230 INPUT B$
4240 CLS
4250 GOTO 1210
5000 REM
5001 REM RIFIUTO PUNTATA
5002 REM
5010 POKE 16418,0
5020 PRINT AT 21,0;"PUNTATA OLTR
E IL TUO CAPITALE   RIPETILA - N
ON SI FA CREDITO"
5030 FOR K=1 TO 50
5040 NEXT K
5050 PRINT AT 21,0;M$;M$
5060 POKE 16418,2
5070 GOTO 1560
5100 REM
5101 REM USCITA ZERO
5102 REM
5120 LET DFA=DF+266
5120 LET DFB=DFA+33
5130 PRINT "0 (ZERO)"
5140 GOTO 3150
5200 REM
5201 REM PASSAGGIO DEL TURNO
5202 REM
5210 PRINT AT 21,0;"VUOI SALTARE
 IL TURNO?"
5220 INPUT B$
5230 GOTO 1520+(230 AND CODE B$=
56)
5500 REM
5501 REM CONTROLLO VINCITA
5502 REM
5510 IF N(I)=NU THEN LET V=1
5520 RETURN
5540 IF N(I)=NU OR N(I)+1=NU THE
N LET V=1
5550 RETURN
5570 IF N(I)<=NU AND N(I)+2)=NU
THEN LET V=1
5580 RETURN
5600 IF N(I)=NU OR N(I)+1=NU OR
N(I)+3=NU OR N(I)+4=NU THEN LET
V=1
5610 RETURN
5630 IF N(I)<=NU AND N(I)+5)=NU
THEN LET V=1
5640 RETURN
5660 IF NU>24 AND N(I)=41 OR NU<
25 AND NU>12 AND N(I)=50 OR NU<1
3 AND N(I)=53 AND NU<0 THEN LET
V=1
5670 RETURN
5690 IF N(I)-INT (N(I)/3)*3=NU-I
NT (NU/3)*3 THEN LET V=1
5700 RETURN
5750 IF NU-INT (NU/2)*2=T(I)-INT
 (T(I)/2)*2 THEN LET V=1
5760 RETURN
5780 IF NU<19 AND T(I)=10 THEN L
ET V=1
5790 RETURN
5810 IF NU>18 AND T(I)=11 THEN L
ET V=1
5820 RETURN
5870 IF PEEK (16513+NU)=PEEK (16
513+T(I)) THEN LET V=1
5880 RETURN
5900 SCROLL
5910 PRINT G$(I);" NON HAI GIOCA
TO"
5920 SCROLL
5930 PRINT TAB 11;"RIMANI CON £
";S(I)
5940 GOTO 4190
5950 SCROLL
5960 PRINT G$(I);" E' USCITO LO
0, LA"
5970 SCROLL
5980 PRINT "TUA PUNTATA RIMANE I
NVARIATA"
5990 GOTO 4190
6000 REM
6001 REM INSERIMENTO TIPO
6002 REM PUNTATA
6003 REM
6100 PRINT AT 21,0;"SU QUALE NUM
ERO?"
6110 INPUT N(I)
6120 IF N(I)<0 OR N(I)>36 THEN G
OTO 6110
6130 PRINT AT 19,25;"SU ";N(I)
6140 RETURN
6200 PRINT AT 21,0;"PRIMO NUMERO
 DEL CAVALLO:"
6210 INPUT N(I)
6220 IF N(I)<1 OR N(I)>35 THEN G
OTO 6270
6230 IF N(I)-INT (N(I)/3)*3=0 TH
EN GOTO 6270
6250 PRINT AT 19,24;N(I);" - ";N
(I)+1
6260 RETURN
6270 PRINT AT 21,0;"CAVALLO NON
POSSIBILE - RIPETI"
6280 GOTO 6210
6300 PRINT AT 21,0;"QUALE TERZIN
A? (PRIMO NUMERO)"
6310 INPUT N(I)
6320 IF N(I)<1 OR N(I)>34 THEN G
OTO 6370
6330 IF N(I)-INT (N(I)/3)*3<>1 T
HEN GOTO 6370
6350 PRINT AT 20,13;"DI ";N(I);"
 - ";N(I)+1;" - ";N(I)+2
6360 RETURN
6370 PRINT AT 21,0;"TERZINA NON
POSSIBILE - RIPETI"
6380 GOTO 6310
6400 PRINT AT 21,0;"QUALE QUADRA
TO?..(PRIMO..NUMERO)"
6410 INPUT N(I)
6420 IF N(I)<1 OR N(I)>32 THEN G
```

```
OTO 6470
6430 IF N(I)-INT (N(I)/3)*3=0 TH
EN GOTO 6470
6450 PRINT AT 20,13;"DI ";N(I);"
 ";N(I)+1;" ";N(I)+3;" ";N(I)+4
6460 RETURN
6470 PRINT AT 21,0;"QUADRATO NON
 POSSIBILE - RIPETI"
6480 GOTO 6410
6500 PRINT AT 21,0;"QUALE SESTIN
A? (PRIMO NUMERO)"
6510 INPUT N(I)
6520 IF N(I)<1 OR N(I)>31 THEN G
OTO 6570
6530 IF N(I)-INT (N(I)/3)*3<>1 T
HEN GOTO 6570
6550 PRINT AT 20,13;"DA ";N(I);"
 A ";N(I)+5
6560 RETURN
6570 PRINT AT 21,0;"SESTINA NON
POSSIBILE - RIPETI"
6580 GOTO 6510
6600 PRINT AT 21,0;"QUALE DOZZIN
A? (P/M/D)"
6610 INPUT B$
6620 IF B$<>"P" AND B$<>"M" AND
B$<>"D" THEN GOTO 6610
6630 PRINT AT 19,25;B$
6640 LET N(I)=CODE B$
6650 RETURN
6700 PRINT AT 21,0;"QUALE COLONN
A? (34/35/36)"
6710 INPUT N(I)
6720 IF N(I)<34 OR N(I)>36 THEN
GOTO 6710
6730 PRINT AT 19,25;"DEL ";N(I)
6740 RETURN
8000 REM
8001 REM ISTRUZIONI
8002 REM
8010 CLS
8020 PRINT TAB 9;"R O U L E T T E"
8030 PRINT ,,TAB 5;"ISTRUZIONI P
ER IL GIOCO"
8040 PRINT ,,,,"POSSONO PARTECIP
ARE 10 GIOCATORIAL MASSIMO"
8045 PRINT ,,"A CIASCUNO VIENE A
SSEGNATO INI- ZIALMENTE UN CAPIT
ALE DI £ 1000"
8050 PRINT ,,"AL TURNO INSERIRE
L'IMPORTO DEL-LA PUNTATA, IL TIP
O ED I RELATI-VI NUMERI SCELTI"
8060 PRINT ,,"NON VI E' ALCUN LI
MITE ALL'IM- PORTO DELLA PUNTAT
A SALVO IL CA-PITALE MASSIMO A D
ISPOSIZIONE"
8070 PRINT AT 21,0;"PREMERE NEWL
INE PER CONTINUARE"
8080 INPUT B$
8090 CLS
8100 LET L=USR 16562
8110 PRINT AT 16,0;"SE ESCE LO 0
, LE PUNTATE PAGATE ALLA PARI RE
STANO SUL TAVOLE E SE COL GIOCO
 SUCCESSIVO ESCONO VENGONO PAGA
TE"
8120 PRINT AT 21,0;"NEW LINE PER
CONTINUARE"
8130 INPUT B$
8140 CLS
8150 PRINT ;"RICORDARSI CHE LO Z
X NON FA CRE-DITO E QUANDO AVRET
E PERSO TUTTOSARETE ELIMINATI"
8160 PRINT ,,"BUON DIVERTIMENTO"
8170 PRINT AT 21,0;"NEW LINE PER
INIZIARE"
8180 INPUT B$
8200 GOTO 1090
9000 SAVE "ROULETTE"
9010 RUN
```

Figura 4. *Listato BASIC del programma Roulette. Per l'inserimento della REM alla linea 1 procedere come segue: copiare da questo listato i primi 36 caratteri "R" e "N" esattamente nell'ordine in cui sono e quindi completare la linea con 56 caratteri qualsiasi. Le scritte in campo inverso sono: linea 520 "Rouge"; linea 1050 "Roulette"; linea 1250 "Signori fate il vostro gioco"; linea 3050 "I giochi son fatti" e "NEW LINE per lanciare"; linea 4030 "Aggiornamento capitali"; linea 4220 "Premere NEW LINE per altra manche"; linea 8020 "Roulette"; linea 8070 "NEW LINE per continuare"; linea 8170 "NEW LINE per iniziare".*

do così, alla chiamata della routine, il trasferimento globale del video.

Analizziamo ora la seconda routine in linguaggio macchina che consente l'inversione del video, ed è illustrata nella figura 3.

Con questa routine lo Z80 passa in rassegna ogni singola locazione del Display File e vi aggiunge o sottrae 128 a seconda che il codice del carattere contenutovi sia minore o uguale/maggiore a 128. Per esempio: se nella 1ª locazione del Display File si trova il carattere di codice 38, corrispondente alla lettera "A", vi viene aggiunto 128 in modo che il nuovo carattere abbia codice 166 (38 + 128), corrispondente alla lettera "A" in campo inverso.

Se inseriamo questa routine in un ciclo FOR ... NEXT, la continua inversione del video ha come effetto il lampeggio dello stesso. Naturalmente il numero di ripetizioni del ciclo deve sempre essere pari, altrimenti alla fine del ciclo il video rimane invertito.

Anche questa routine può essere usata solo su parte del video. La locazione che contiene il numero delle linee oggetto della inversione è la quinta della routine; variandone il contenuto con valori da 1 a 24 si ottiene l'effetto sul numero di linee inserite in tale locazione (nel programma roulette 18).

Passiamo ora ad analizzare il listato BASIC della figura 4. Le REM inserite lo dividono in blocchi per renderne più chiara la lettura. Esaminiamo i blocchi individualmente.

3115 - 3180: stampa il numero uscito e provvede al lampeggio dello stesso sul tappeto.

4000 - 4250: aggiorna i capitali dei singoli giocatori.

5000 - 5070: in caso la puntata sia superiore al capitale a disposizione lo ZX lo segnala scherzosamente dicendo che "non si fa credito" e invitando il giocatore a ripeterla.

5200 - 5230: subroutine per la conferma da parte del giocatore che intende saltare il turno di gioco.

5100 - 5140: subroutine per l'uscita dello zero.

5500 - 5880: controlla se il giocatore ha vinto e, nel caso affermativo pone a 1 la variabile V.

5900 - 5990: aggiornamento capitali per chi non ha giocato o ha puntato sulle combinazioni semplici ed è uscito lo "zero".

6000 - 6740: inserimento delle puntate numeriche. Ognuna di essa è contraddistinta da un numero da 1 a

7 (vedasi figura 1 "Istruzioni") e per ogni tipo vi sono le istruzioni alle relative centinaia.
8000 - 8200: subroutine contenente le istruzioni alle quali si accede nella fase iniziale del gioco.

Il programma presenta inoltre un notevole uso dell'operatore logico AND che permette un notevole risparmio di memoria.
Si prenda ad esempio la linea:
1080 GOTO 1080 + (10 AND INKEY$ = "N") + (6920 AND INKEY$ = "S")
che sostituisce le seguenti linee:
1080 IF INKEY$ = "S" THEN GOTO 8000
1085 IF INKEY$ < > "N" THEN GOTO 1080
altro esempio sono le linee 3120 - 3130 - 3140 che sostituiscono ben 6 istruzioni IF e le linee 380 e 1150 che consentono l'incolonnamento a destra di numeri inferiori alle centinaia.

Nonostante l'uso del suddetto operatore il programma occupa quasi 9 Kbyte di memoria per le sole istruzioni.
Si ricorda infine che i numeri neri sono stampati nero su bianco mentre quelli rossi sono stampati in campo inverso in seguito alla mancanza del colore nello ZX81. Per quanto riguarda la stampa del tappeto viene utilizzato il carattere grafico cod. 8 (GRAPHICS SHIFT A).
1: REM iniziale contenente alle prime 36 locazioni i caratteri che identificano il colore del numero uscito. Seguono le due routine in linguaggio macchina illustrate ed i caratteri contenenti il valore delle vincite relative ai vari tipi di puntate possibili. Per l'inserimento della REM procedere come segue: copiare dalla figura 4 i primi 36 caratteri "R" e "N" così come scritti e continuare la linea REM con altri 56 caratteri a caso.
100 - 180: carica nella REM 1 il linguaggio macchina relativo alle due routine utilizzate nel programma per la velocizzazione del video. I codici sono contenuti nella stringa B$, in esadecimale.

G $ (10, 10)	Matrice contenente il nome dei giocatori.
S (10)	Matrice contenente la somma a disposizione dei giocatori.
P (10)	Matrice contenente l'importo della puntata.
T (10)	Matrice contenente il tipo di puntata scelta.
N (10)	Matrice contenente il numero relativo al tipo di puntata scelta.
NG	Numero dei giocatori.
DF	Inizio del Display File.
DFA/DFB	Locazioni che identificano nel Display File il numero uscito (necessarie per il lampeggio dello stesso).
NU	Numero uscito.
C $ (13, 7)	Matrice contenente i tipi di combinazioni possibili.
V	Identificatore della vincita.

Figura 5. *Elenco delle variabili utilizzate.*

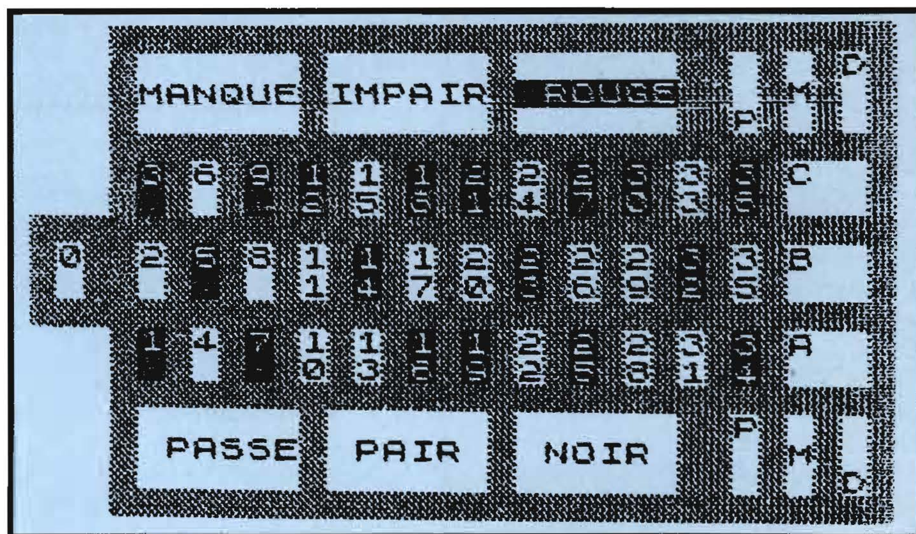


Figura 6. *Immagine del tappeto di gioco come appare sul video. Questa immagine viene richiamata velocemente nel Display File usando la routine in linguaggio macchina della figura 2.*

200 - 410: predispone il video con i tipi di puntate possibili e le relative vincite; il tutto viene salvato nella RAM alta.
500 - 620: viene creato il tappeto di gioco ed anch'esso salvato nella RAM alta.
700 - 790: generazione delle matrici per il gioco; vedasi nella figura 5 "elenco delle variabili" a cosa si riferiscono.
1000 - 1200: inizio gioco che rimanda alla stampa delle istruzioni in caso di richiesta ed introduzione numero dei giocatori con i relativi nomi.
1210 - 1270: presentazione del tappeto di gioco con invio alla puntata.
1500 - 1760: introduzione delle puntate: prima il capitale rischiato, poi il tipo di puntata ed i numeri alla quale si riferisce.

Vi è la possibilità di saltare il turno inserendo 0 alla richiesta del capitale. Prima di accettare la puntata viene richiesta conferma al giocatore.
3000 - 3040: scelta casuale del numero uscito e predisposizione delle variabili per il lampeggio del numero sul tappeto.
3050 - 3100: utilizzo della routine in linguaggio macchina per il lampeggio del tappeto di gioco per dare l'impressione della scelta casuale del numero. ■

PERSONAL SOFTWARE

Type - Writer 1

— Parte prima —

Un potente text editor per il Commodore VIC 20

di *Alessandro Guida*

Il nome dato al programma chiarisce lo spirito di questo lavoro: Type-Writer vuol dire "macchina per scrivere". Infatti, il VIC verrà trasformato in una moderna macchina per scrivere, dotata di numerosi ed utilissimi comandi per facilitare la stesura di pagine di testo.

Cos'è un Text Editor

Un Text Editor è un programma che permette la stesura, correzione, stampa ed archiviazione su memoria di massa di testi.
Il primo vantaggio offerto da un Text Editor è quello di potersi spostare con il cursore per tutto il testo. Diventano, quindi, molto facili le operazioni di correzioni che si ottengono semplicemente portando il cursore sulla parola sbagliata e riscrivendola corretta.
Inoltre, questo tipo di programmi è dotato di tutta una serie di comandi che permettono di alleggerire notevolmente sia il lavoro di composizione che quello di correzione.

Ad esempio, tra le possibilità offerte dai Text Editor c'è sempre l'allineamento automatico a destra del testo, la inserzione e cancellazione di intere linee di testo, e il ritorno a capo automatico con il trasferimento alla riga successiva della eventuale parola incompleta.

```
                    TYPE-WRITER 1

                 TEXT EDITOR PER VIC 20

****************************************************************
*                                                              *
* Programma realizzato dalla:                                  *
*                                                              *
*     STUDIO AG Software  di Alessandro Guida                  *
*                                                              *
*     via Panzani,13  50123 FIRENZE                            *
*                                                              *
*                                   (C) 8-Giugno-1983          *
*                                                              *
****************************************************************


Questo Programma lungi dall'essere considerato un word-
Processor e' stato concepito Per la massima facilita' d'uso.
Percio' e' video orientato, cioe' sul monitor si ha sempre
l'esatto formato di stampa, ed ogni operazione sul testo
viene effettuata aggiornando immediatamente il monitor.
Lo schermo e' una finestra di 22 colonne x 22 linee su un
foglio di 80 x 70.
Di seguito sono riportati tutti i comandi di cui dispone
questo text editor e una spiegazione del loro
funzionamento.
Ricordiamo intanto alcune interessanti particolarita'. La
piu' insolita ma oseremmo dire la piu' sofisticata e' la
possibilita' di passare in qualsiasi momento in modo
calcolatore per eseguire operazioni aritmetiche. Con la
semplice pressione di un tasto si torna al testo o
addirittura si puo' avere l'introduzione automatica del
risultato nel punto in cui si era lasciato il cursore.
Altre particolarita' sono la giustificazione del testo a
```

Figura 1. *Esempio di testo composto con il programma presentato in questo articolo.*

È chiaro, quindi, che non sarà più necessario buttare via decine di fogli di carta per errori di sintassi poichè si potrà controllare accuratamente il testo e correggerlo prima di inviarlo alla stampante.

Implementazione sul VIC

Il VIC impone delle limitazioni, dovute sia allo schermo da 22 colonne e sia alla poca memoria disponibile.

Type - Writer 1

Il programma è stato scritto per il VIC con una espansione da 16 Kbyte ed una stampante. Come memoria di massa è possibile utilizzare sia il registratore che il lettore floppy disk.
Anche in questa configurazione, però, sono necessari alcuni compromessi. Primo, non è possibile la visualizzazione di tutto il testo sullo schermo ma solo di una parte detta "finestra". Questa finestra è gestita automaticamente dal programma che provvede a spostarla quando il cursore giunge al limite dello schermo in una qualsiasi delle quattro direzioni.

Il secondo compromesso è dovuto ai 16 Kbyte di RAM che sono pochini per contenere il programma e i testi contemporaneamente. Perciò si è limitato il testo ad una pagina per volta (80 colonne per 70 righe). Va detto, però, che la gestione di più pagine da disco o da nastro è molto semplice e veloce.

Il programma

Il programma, lungi dall'essere considerato un word-processor, è stato concepito per la massima facilità d'uso.
Perciò è video-orientato, cioè sul monitor si ha sempre l'esatto formato di stampa, ed ogni operazione sul testo viene effettuata aggiornando immediatamente il video.
Lo schermo è una finestra di 22 colonne x 22 linee su un foglio di 80 x 70.
Di seguito riporteremo tutti i comandi di cui dispone "Type-Writer 1" ed una spiegazione del loro funzionamento.

Ricordiamo, intanto, alcune interessanti peculiarità di questo programma.

```
          GGL Software

          via G.A.Milone,18

     72021 Francavilla F.          P.Iva 0123456789012345

                                        Spett. Personal Software
                                               via Rosellini,12
                                         20100 MILANO

                                               P.Iva 111111111111111

     Fattura n. **  del **/**/****

     Rif. Bolla c. n. **  del **/**/****

                                          Costo         Costo
     Quant.  Descrizione                 Unitario       Totale
     ----------------------------------------------------------
        *    ************************     *****         ******
        *    ************************     *****         ******
                                                     -----------
                                              TOTALE *******
                                                     -----------
                                             IVA 18% ******
                                                     -----------
                                        TOT. FATTURA *******
     Pagamento:
```

Figura 2. *Esempio di maschera utilizzabile per l'emissione di piccole fatture. Il completamento delle varie voci è facilitato dalla possibilità di eseguire i calcoli senza mai abbandonare il Text-Editor.*
Il tasto "↑" permette di riportare direttamente il risultato nel testo.

COMANDI DA TASTIERA

CRSR UP, DOWN RIGHT, LEFT	Muovono il cursore e una volta raggiunto uno dei bordi provocano lo scrolling della finestra video.
SHIFT	Attiva la seconda funzione del tasto premuto contemporaneamente. Per le lettere si hanno le maiuscole.
RETURN	Ritorno a capo del rigo seguente.
HOME	Riporta il cursore all'angolo in alto a sinistra.
CLR	Pulisce completamente la pagina video.
DEL	Cancella il carattere che precede il cursore spostando tutti i caratteri seguenti fino al termine della linea.
INST	Inserisce uno spazio spostando a destra i caratteri a partire da quello sotto il cursore. Nello spingere a destra i caratteri vengono eliminati gli spazi superflui per cercare di non superare il margine destro imposto.

PERSONAL SOFTWARE

Type - Writer 1

La più insolita, ma oseremmo dire la più sofisticata è la possibilità di passare in qualsiasi momento in "modo calcolatore" per eseguire operazioni aritmetiche. Con la semplice pressione di un tasto si torna al testo o, addirittura, si può avere l'introduzione automatica del risultato nel punto in cui si era lasciato il cursore. Altre particolarità sono la giustificazione del testo a destra (non obbligatoria!), la possibilità di definire dei margini entro i quali comporre il testo e, infine, l'avere conservato intatte le funzioni originali di editing della Commodore.
Come abbiamo detto, il programma gira sul VIC con espansione RAM da 16 Kbyte. La memoria è così suddivisa:

\$1000-\$11FF	Area Schermo
\$1200-\$3FFFF	Area BASIC
\$4000-\$5600	Pagina Testo
\$5700-\$5FFF	Routine in L.M.

Gli usi di questo programma sono innumerevoli. Si possono scrivere lettere, testi, piccoli manuali o anche stampare fatture come si può vedere in figura 2. Quest'ultimo compito è facilitato dalla possibilità di tenere sempre su disco la maschera della fattura e dal fatto di poter eseguire calcoli senza abbandonare il text-editor.

Uso del programma

L'uso di "Type-writer 1" è estremamente semplice. Dato LOAD "TYPE-WRITER 1" (,8 se si utilizza il disco) viene caricata in memoria la parte contenente le routine in linguaggio macchina.
Battuto il RUN, le routine vengono allocate in memoria, e il programma provvede da solo a caricare la restante parte.
Subito dopo appare la richiesta di alcuni parametri. Se non vogliamo modificarli basta premere RETURN.

COMANDI DIRETTI

Questi comandi vanno dati premendo contemporaneamente il tasto indicato e il tasto "CTRL".

Tasto	Funzione	Descrizione
/(?)	HELP	Fornisce in ogni momento il riepilogo dei comandi a disposizione, senza perdere il testo in memoria.
A	AUTO	Esegue l'allineamento del testo con il margine destro.
B	BLOC	Questo comando va dato con il cursore sulla prima colonna. Compare una "B" in reverse che impedisce l'esecuzione dell'allineamento a destra sulla stessa linea.
C	CALC	Entra in MODO CALCOLATORE (vedi).
D	DEL	Cancella tutti i caratteri della riga dal cursore in poi.
F	FIND	Cerca nel testo una stringa lunga al massimo 10 caratteri. La richiesta del testo da cercare compare sull'ultima riga in basso dello schermo. Si scrive il testo e si batte RETURN. Se il testo non viene rintracciato all'interno della pagina, appare il messaggio "NOT FOUND" con la domanda se si vuole eseguire un'altra ricerca. Se invece la ricerca ha risultato positivo il cursore si sposta sulla prima ricorrenza del testo nella pagina. Per proseguire nella ricerca è sufficiente premere ancora CTRL + F.
H	HIGH	Seleziona i caratteri in doppia larghezza. Questo comando viene accettato solo se il cursore si trova sulla prima colonna, dove viene riportata una D in reverse per conferma. Il computer considererà, poi, doppi i primi 40 caratteri appartenenti alla stessa linea, ignorando, in fase di stampa, tutti gli altri fino al termine della riga.
I	INS	Inserisce una linea vuota sotto il cursore facendo scivolare tutte le linee seguenti in basso di una posizione.
M	MEM	Entra in MODO GESTIONE MEMORIA DI MASSA (vedi).
P	PRINT	Stampa su carta una copia esatta della pagina di testo composta.
S	SEQ	Stampa su carta una sequenza di pagine caricate dal disco.
T	TRANS	Ricopia una riga di testo da un punto ad un altro. Basta posizionare il cursore sulla linea da trasferire e premere i tasti CTRL e T. Per rammentare che la linea è pronta per essere trasferita apparirà sull'ultima riga di schermo una T in reverse. Ora, in qualunque momento vengono premuti nuovamente i tasti CTRL e T, la riga viene ricopiata sulla linea dove si trova il cursore.
U	UP	Cancella la linea sotto il cursore forzando le linee seguenti a risalire di una posizione.
V	VIEW	Permette di osservare la pagina con uno scrolling molto veloce. I tasti cursore spostano la finestra nelle quattro direzioni. La pressione di qualsiasi altro tasto ritorna all'editing normale.
@	PAR	Ritorna alla definizione dei parametri iniziali.
1,2,3	TAB	Spostano il cursore, direttamente ad un punto della linea stabilito all'inizio del programma o con il comando CTRL + @ .



Type - Writer 1

Le informazioni richieste sono:
TAB 1-2-3. Sono tre posizioni sullo schermo alle quali si può saltare direttamente premendo il tasto CTRL più il tasto 1 o 2 o 3.
Margine sinistro-destro. Sono i margini entro i quali deve essere contenuto il testo. Il testo che vedete in figura 1, ad esempio, usa margine sinistro = 10 e margine destro = 70.
Numero righe per pag. Indica al programma di quante righe è composto il foglio. Tale valore è normalmente 66 o 72.
A questo punto appaiono sull'ultima linea in basso dello schermo, le coordinate del cursore, e la finestra sul testo da comporre.

Da questo momento, si può cominciare a digitare il testo desiderato.
Inizialmente lo schermo appare pieno di caratteri casuali. Questo è dovuto al fatto che all'inizio non viene ripulita la pagina di schermo per permettere, in caso di errore, di riprendere il programma senza perdere il testo già scritto.

I comandi

I comandi sono di tre tipi.
1) I comandi di tastiera. Sono funzioni che vengono attivate semplicemente premendo il tasto corrispondente sulla tastiera.
2) I comandi diretti. Sono i comandi che vengono eseguiti premendo il tasto CTRL più il tasto corrispondente al comando scelto.
3) I comandi indiretti. Sono i comandi a cui si acccede per mezzo di un'altro comando.

I file testo e i file sequenziali

È possibile trattare anche piccoli file sequenziali, lunghi non più di 5600 byte. I file sequenziali sono rappresentati sullo schermo normalmente con i vari record separati da un RETURN (codice ASCII =

COMANDI INDIRETTI

A questi comandi si accede con uno dei due comandi diretti visti prima CTRL + C o CTRL + M. Per i comandi indiretti non è necessaria la pressione del tasto CTRL.

	MODO CALCOLATORE	
X.*	MOLT	Segni validi per moltiplicare due numeri.
+	ADD	Somma.
—	SUB	Sottrazione.
:, /	DIV	Ambedue validi per indicare la divisione.
%	PERC	Serve a calcolare la percentuale, indicata dal secondo numero, del primo. Se per esempio vogliamo calcolare il 20% di 160 si dovrà impostare: 160 % 20 = e il risultato sarà 32
=	RES	Calcola il risultato.
Q	QUIT	Ritorna al Text-Editor.
!	ENTER	Ritorna al Text-editor inserendo il risultato del calcolo nel punto del testo in cui era stato lasciato il cursore.
	MODO GESTIONE MEMORIA MASSA	
L	LOAD	Carica dal floppy o dal registratore una pagina di testo. Attenzione, perchè eventuali testi già in memoria vanno persi.
S	SAVE	Registra nella memoria di massa una pagina di testo dopo aver specificato se si tratta di un testo o di un file sequenziale.
R	REN	Permette di cambiare il nome assegnato ad una pagina già registrata.
F	FORM	Esegue la formattazione di un dischetto nuovo.
D	DEL	Cancella dal dischetto una pagina già registrata.
$	DIR	Visualizza il contenuto di un dischetto.
Q	QUIT	Torna al Text-Editor.

DIFFERENZA TRA TEXT-EDITOR E WORD-PROCESSOR

Probabilmente molti di voi avranno sentito parlare dei programmi di word-processing, e si staranno chiedendo la differenza che intercorre tra questi e un text-editor.
In effetti, questi due termini sono molto simili e indicano entrambi una categoria di programmi volti all'elaborazione automatica dei testi.
In particolare, il word-processor, come dice lo stesso nome, scende fino al livello della singola parola. Rappresenta, quindi, un passo in avanti rispetto al Text-Editor.
È, infatti, sempre dotato di opzioni che gli permettono la divisione sillabica delle singole parole, o la ricerca e sostituzione di una parola con un'altra. Alcuni di questi word-processor sono, addirittura, in grado di controllare l'esatta sintassi delle parole.
Inoltre, sono tutti in grado di collegarsi con programmi di archiviazione per l'introduzione automatica, nei testi elaborati, di dati e informazioni provenienti da detti archivi.
In pratica tali programmi trovano applicazione (anche a causa del loro costo non trascurabile) negli uffici, dove la velocità di elaborazione e la necessità di eseguire stampe di circolari personalizzate sono esigenze molto sentite.

13) rappresentato con una freccia in reverse. Nell'editing di questi file il return va introdotto battendo SHIFT + RETURN.
La fine del file si indica con il carattere freccia in su.

Nell'operazione di SAVE, naturalmente va specificato se si tratta di un file di testo o di un file sequenziale.

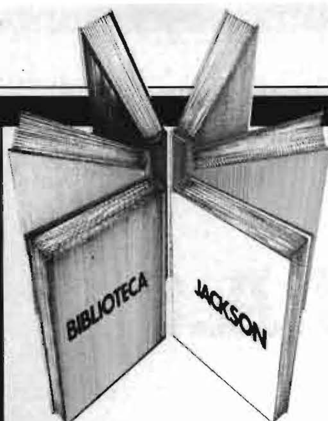
Conclusioni

Come avrete certamente potuto apprezzare, il Type-Writer 1, pur rimanendo un programma studiato per il piccolo VIC 20, non sfigura al confronto di text-editor per macchine più grosse. Anzi, può vantare delle caratteristiche che nessun altro programma del genere possiede (vedi il modo calcolatore).
Lo sforzo principale è stato comunque quello di fare un programma coerente con le dimensioni del VIC e delle periferiche più economiche di cui dispone.

La prossima puntata pubblicheremo tutti i listati, e le notizie relative. Sarà disponibile anche il programma su dischetto. ■

COMANDI DIRETTI (PREMERE CTRL)					
/(?)	IST	H	HIGH	U	UP
A	AUTO	I	INS	V	VIEW
B	BLOC	M	MEM	@	PAR
C	CALC	P	PRINT	1	TAB1
D	DEL	S	SEQ	2	TAB2
F	FIND	T	TRANS	3	TAB3
COMANDI INDIRETTI (SENZA IL CTRL)					
$	DIR	L	LOAD	S	SAVE
D	DEL	Q	QUIT	!	ENTER
F	FORM	R	REN		

Lista dei principali comandi del Type-Writer.



Libri firmati JACKSON



VOI E IL VOSTRO COMMODORE 64
Un esaudiente vademecum sulla programmazione in BASIC dal Personal ad oggi tra i più diffusi.
Facile, brillante ricco di programmi verificati, questo è un prezioso volume sia per i neofiti che per gli utilizzatori più esigenti.
256 pag. **L. 22.000**
Cod. **347 B.**

PROGRAMMAZIONE DELLO ZX SPECTRUM
Aggiungete suono e colore ai vostri programmi, scoprite lo SPECTRUM negli affari e nell'istruzione, giocate e imparate a scrivere i giochi, disegnate figure in 3 dimensioni.
212 pag. **L. 18.000**
Cod. **531 D.**

APPLE MEMO
Sintassi dei comandi, codici dei caratteri, messaggi di errore, linguaggio macchina, indirizzi utili.....
Un libro destinato a stare in permanenza a fianco del vostro Apple.
146 pag. **L. 15.000**
Cod. **340 H.**

BASIC SU APPLE
programmi in pochi minuti
65 programmi pronti che vi risolveranno problemi che vanno dalla "economia domestica", alle applicazioni commerciali, ai calcoli statistici, alla creazione degli archivi.
184 pag. **L. 14.000**
Cod. **532 H.**

APPLE TUTTO FARE
Collegamenti e progetti
Questo libro è stato scritto per chiunque voglia capire come l'APPLE e gli altri home computer, possano essere interfacciati con il mondo esterno.
208 pag. **L. 18.000**
Cod. **334 D.**

PET/CBM GUIDA ALL'USO VOL. 1 e VOL. 2
È la versione italiana del famosissimo testo americano: "PET/CBM Personal Computer Guide" ed è presentato in due volumi data l'ampiezza e la profondità degli argomenti trattati. In questo manuale troverete tutto ciò che è necessario sapere sui calcolatori COMMODORE.
VOL 1 250 pag. **L. 20.000**
cod. **332 P**
VOL 2 282 pag. **L. 22.500**
cod. **333 P**

La Biblioteca che fa testo



GRUPPO EDITORIALE JACKSON

Attenzione compilare per intero la cedola
ritagliare (o fotocopiare) e spedire in busta chiusa a:
GRUPPO EDITORIALE JACKSON
Divisione Libri
Via Rosellini, 12 - 20124 Milano

CEDOLA DI COMMISSIONE LIBRARIA

VOGLIATE SPEDIRMI

n° copie	codice	Prezzo unitario	Prezzo totale
		Totale	

☐ Pagherò contrassegno al postino il prezzo indicato più **L. 2000 per contributo fisso spese di spedizione**

Condizioni di pagamento con esenzione del contributo spese di spedizione:

☐ Allego assegno della Banca ____________

n° ____________

☐ Allego fotocopia del versamento su c/c n. 11666203 a voi intestato

☐ Allego fotocopia di versamento su vaglia postale a voi intestato

Nome
Cognome
Via
Cap Città Prov.
Data Firma

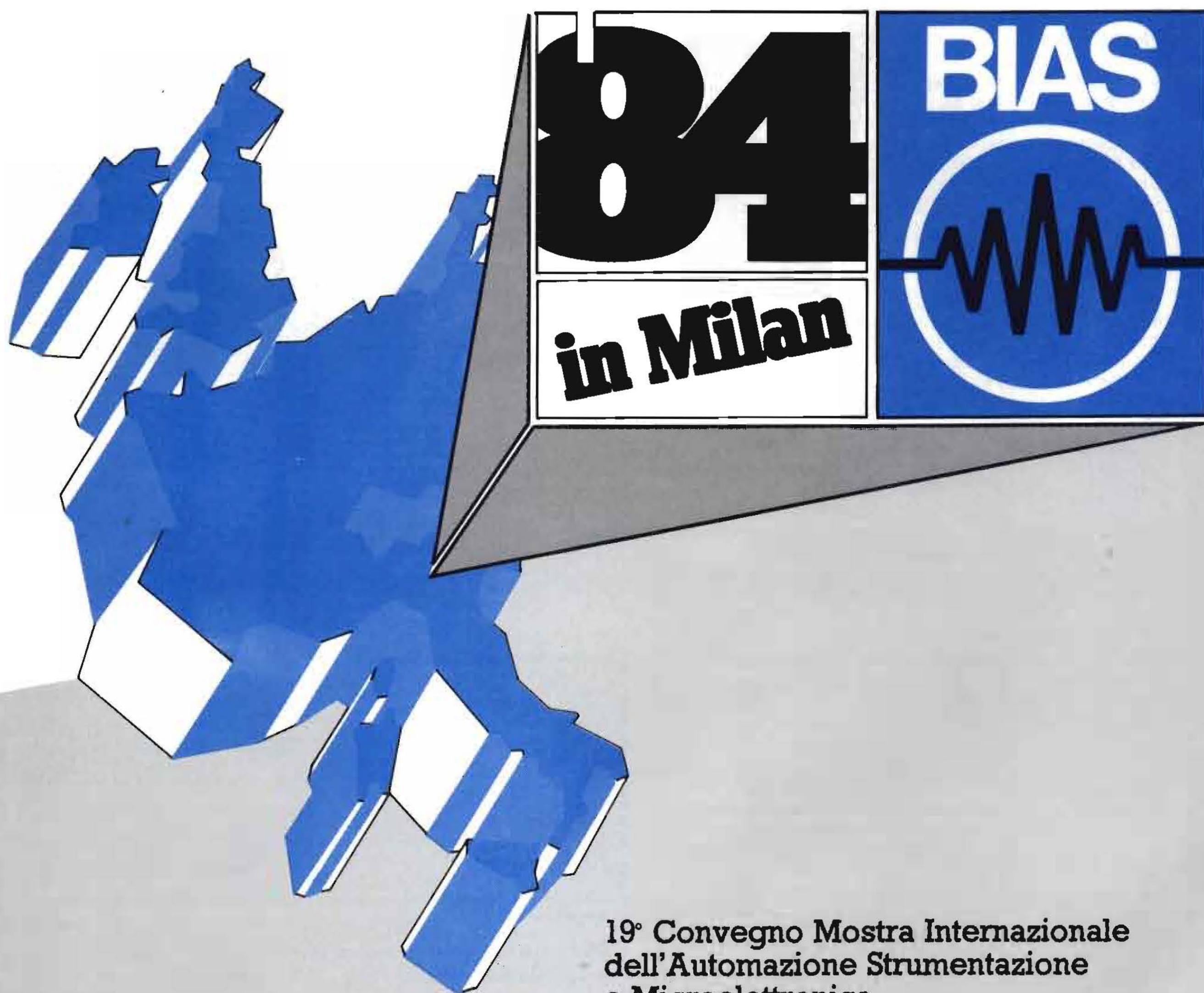
Spazio riservato alle Aziende. Si richiede l'emissione di fattura
Partita I.V.A. |_|_|_|_|_|_|_|_|_|_|_|
ORDINE MINIMO L. 50.000

Esposizioni Internazionali dell'Automazione

...1982 Parigi "MESUCORA"... 1983 Düsseldorf "INTERKAMA"

1984 MILANO - B.I.A.S.

Solo il BIAS nel 1984 in Europa presenta l'Automazione e la Microelettronica



19° Convegno Mostra Internazionale dell'Automazione Strumentazione e Microelettronica

Fiera di Milano
29 novembre - 4 dicembre 1984

- Sistemi e Strumentazione per l'Automazione la regolazione ed il controllo dei processi Robotica, sensori e rilevatori
- Apparecchiature e Strumentazione per laboratorio, collaudo e produzione
- Componentistica, sottoassiemi periferiche ed unità di elaborazione
- Micro, Personal Computer, Software e accessori

E.I.O.M. Ente Italiano Organizzazione Mostre
Segreteria della Mostra
Viale Premuda 2
20129 Milano
tel. (02) 796096/421/635 - telex 334022 CONSEL

in concomitanza con la 8ª RICH e MAC '84



Schermo Apple - C 64 e la ragnatela

Un programma per convertire facilmente software da Apple a Commodore e viceversa e un gioco che ne esemplifica l'uso

di *Claudio Poma*

Il programma schermo Apple e C 64 del listato 1, serve a tradurre i dati riferiti agli schermi dei due computer.

Come sapranno tutti coloro che hanno provato a convertire un programma per Apple II sul loro C 64, o viceversa, il problema che presenta le maggiori difficoltà (non tanto tecniche, quanto in perdita di tempo) è quello di ottenere una precisa visualizzazione dello schermo: basta una colonna in più o in meno, una minima discrepanza nelle POKE e se ne vedono veramente di tutti i colori.

Ora, una volta fatto girare questo programma scritto su un Apple, ma che gira perfettamente anche sul C 64 senza bisogno di alcuna modifica, tutti questi problemi non faranno più perdere tempo prezioso, oltre a dimostrare come i computer possano essere utilizzati per superare almeno alcune differenze che li caratterizzano.

Come funziona

Una volta dato il RUN, il programma chiede quale sia la macchina di cui si hanno i dati: attenzione quindi perchè non bisogna dire qual'è il proprio computer, ma quello per cui è stato scritto il programma che si vuole convertire.

Dopo la verifica di quanto detto (linee 60-90) il programma passa alla parte pratica. Se dovete tradurre dei dati dell'Apple in quelli del C 64, vi sarà chiesto se tali dati sono espressi in POKE o VTAB-HTAB; tutti i dati successivamente introdotti verranno convertiti nei corrispondenti valori schermo dell'altra macchina e per quanto riguarda Apple anche la POKE individuata dalla coppia VTAB-HTAB richiesta.

In questo modo il programma può essere utilizzato anche solo per convertire un VTAB-HTAB in POKE sempre per Apple, cosa che ogni tanto può servire. Non sono trattati i valori di riga e colonna del C 64 non per negligenza, ma solo per non allungare il programma con operazioni elementari: infatti per ottenere tali valori basta diminuire di 1 i valori di VTAB e HTAB che sono evidenziati in ogni punto del programma. Questo perchè la prima riga e la prima colonna assumono nell'Apple il valore 1, mentre nel C 64 il valore 0. A parte questo nessuno vi impedisce di effettuare, in modo molto semplice, le aggiunte necessarie.

Se siete invece in possesso dei dati del C 64 e volete convertirli negli equivalenti dell'Apple II, vi sarà chiesto se tali dati riguardano la memoria schermo o colore riportando i valori limite (linee 310-370); quindi dovrete battere i dati che possedete e immediatamente avrete la traduzione per l'Apple e dato che lo schermo di tale computer ha 24 linee contro le 25 del C 64, se la POKE del Commodore è posta sull'ultima riga, il programma vi avvisa della cosa con la linea 455. Lo stesso avviene quando all'introduzione di dati POKE dell'Apple questi corrispondono a POKE non visualizzate (linea 660), anche se tale possibilità dovrebbe essere solo teorica nella conversione di un programma.

Il programma non ha la pretesa di risolvere le differenze esistenti tra sistemi diversi e quel poco che fa può senz'altro essere potenziato da ogni lettore in riferimento alle esigenze personali; nonostante ciò resta un buon esempio delle possibilità non ancora sfruttate in questa direzione e chissà che in un tempo non molto lontano non ci si debba preoccupare più di tanto in merito al computer cui è destinato un programma ...

La ragnatela

Il programma in questione è un gioco e, più precisamente, un labirinto. L'argomento è certamente sfruttato, ma non si tratta di mangiare solo puntini: ci sono tesori, il fantasma del guardiano che insegue passando attraverso i muri, un tempo da rispettare, bonus ... il tutto sempre ben evidenziato sullo schermo in ogni momento del gioco e poi il programma si presenta come complemento ideale a Schermo Apple E C 64 e come buon esercizio di traduzione per i possessori di C 64. Infatti il listato 2 è la versione per Apple II del gioco: gli utenti del C 64 possono facilmente tradurlo usando appunto il programma precedente.

Descrizione del programma

Dopo la definizione delle variabili viene stampato il labirinto, quindi la linea 70 manda alla subroutine che posiziona i tesori scegliendo a caso tra trenta punti predefiniti con delle DATA. La linea 80, seguendo lo stesso procedimento, piazza le trap-



pole. Il gioco inizia: le linee 100-107 danno le informazioni aggiornate in merito a punti, tempo, record, N° giro, N° tesori rimasti da trovare per terminare il percorso, punti che si perderebbero cadendo in una trappola e, infine, punti bonus che si vincerebbero terminando in quel momento il percorso.

La linea 109 cancella i simboli del giocatore (★) e del guardiano (X). Per quest'ultimo si è usata una variabile (L$) in modo da permettere il passaggio tra i muri e su tesori e trappole senza che questi vengano cancellati.

Dalla 110 alla 135 viene definito il movimento del giocatore: la PEEK(—16384) ricorda l'ultimo tasto premuto; mentre da 140 a 170 viene definita la posizione del guardiano tenendo in considerazione la posizione del giocatore (150-160).

Nella subroutine 1000-1100 le linee da 1000 a 1040 definiscono la posizione del guardiano mentre dalla 1050 alla 1080 viene definito il carattere che dovrà rappresentare il guardiano quando questo si sarà spostato in una nuova posizione. La linea 180 manda alla routine di fine partita qualora le posizioni del giocatore e del guardiano coincidano, mentre se questo non avviene il programma continua leggendo il carattere della nuova posizione del giocatore e agendo di conseguenza a seconda che si tratti del muro, di tesori o di trappole. Quindi il programma ricomincia a calcolare una nuova mossa.

Conclusioni

Il programma non ha nessuna pretesa di originalità e inoltre può essere arricchito con altre varianti che renderebbero il gioco più vario ed impegnativo: da una diversa forma del labirinto per ogni fase del gioco ad un numero crescente di guardiani e ad una loro maggiore precisione nell'inseguire il giocatore. La sua utilità sta, invece, nella facilità con cui può essere convertito per il C 64 per mezzo dell'aiuto del programma Schermo Apple e C 64 che è essenziale per la traduzione delle linee DATA, mentre il resto è praticamente identico a parte le linee da 1000 a 1040 e la routine 500 più le linee 182-183 che sono semplificabili nella formula generale POKE SCHERMO = 1024+Y+40★X. ■

Listato 1. *Programma Schermo Apple e C 64. Può essere utilizzato sia dai possessori di Apple che da quelli di Commodore.*

```
10  HOME : REM  C64 10 ?"<SHIFT-CLR/HOME>"
20  PRINT : PRINT : PRINT "HAI I DATI DI UN APPLE ][ O DI UN C 64 ?"
25  PRINT "(A/C) ? ";
30  GET A$: IF A$ = "A" THEN B$ = "APPLE ][": GOTO 60
40  IF A$ = "C" THEN B$ = "C 64": GOTO 60
50  GOTO 30
60  PRINT : PRINT : PRINT "E' VERO CHE HAI I DATI DI UN "B$" ?"
65  PRINT "(S/N) ? ";
70  GET A$: IF A$ = "N" THEN 10
80  IF A$ = "S" AND B$ = "APPLE ][" THEN 100
85  IF A$ = "S" AND B$ = "C 64" THEN 300
90  GOTO 70
100  HOME
101  PRINT : PRINT : PRINT "Ora dimmi se usi le POKE (1024-2047) o  VTAB e HTAB.": PRINT "(P/T) :";
102  INPUT AP$
103  IF AP$ = "P" THEN 600
104  IF AP$ = "T" THEN  HOME : GOTO 110
105  GOTO 102
110  PRINT : PRINT : PRINT "ORA DIMMI I DATI DELL'APPLE ][ :"
120  PRINT : PRINT "COLONNA (HTAB) : ";: INPUT HT
130  PRINT : PRINT "RIGA    (VTAB) : ";: INPUT VT
140  PO = 1024 + (HT - 1) + 40 * (VT - 1)
150  CO = 55296 + (HT - 1) + 40 * (VT - 1)
155  X = VT:Y = HT: GOSUB 500:PA = Q
160  PRINT : PRINT : PRINT "HTAB"HT" E VTAB"VT" DELL'APPLE ]["
165  PRINT "CORRISPONDONO A POKE"PA: PRINT : PRINT "E ";
170  PRINT "CORRISPONDONO NEL C 64 A :"
180  PRINT "- SCHERMO  POKE  "PO
190  PRINT "- COLORE   POKE  "CO
200  PRINT : PRINT : PRINT "ANCORA ? (S/N) ";
210  GET A$: IF A$ = "N" THEN  END
220  IF A$ = "S" THEN 100
230  GOTO 210
300  HOME
310  PRINT : PRINT : PRINT "ORA DIMMI SE USI I DATI DELLA MAPPA"
320  PRINT "- DELLO SCHERMO   (1024-2023)"
330  PRINT "- O DEI COLORI  (55296-56295)"
```

PERSONAL SOFTWARE

Schermo Apple - C 64 e la ragnatela

Seguito listato 1.

```
340 PRINT : PRINT "(S/C)";: INPUT
    MS$
350 IF MS$ = "S" THEN MS = 1024:
    GOTO 380
360 IF MS$ = "C" THEN MS = 55296
    : GOTO 380
370 GOTO 340
380 HOME
390 PRINT : PRINT : PRINT "ORA D
    IMMI I DATI DEL C 64"
400 PRINT : PRINT "POKE ";: INPUT
    PO
410 VT =  INT ((PO - MS) / 40) +
    1
420 HT = PO - MS - 40 * (VT - 1) +
    1
425 IF VT < 25 THEN X = VT:Y = H
    T: GOSUB 500:PA = Q
430 PRINT : PRINT : PRINT "A POK
    E"PO" DEL C 64 CORRISPONDONO
    "
440 PRINT : PRINT "VTAB"VT
450 PRINT "HTAB"HT
455 IF VT < 25 THEN  PRINT "POKE
    "PA
460 PRINT : PRINT : PRINT "ANCOR
    A ? (S/N) ";
470 GET A$: IF A$ = "N" THEN  END

480 IF A$ = "S" THEN 380
490 GOTO 470
500 IF X = > 1 AND X = < 8 THEN
    Q3 = 1023:XX = X
510 IF X = > 9 AND X = < 16 THEN
    Q3 = 1063:XX = X - 8
520 IF X = > 17 AND X = < 24 THEN
    Q3 = 1103:XX = X - 16
530 Q = Q3 + Y + 128 * (XX - 1)
540 RETURN
600 HOME
610 PRINT : PRINT : PRINT "Ora d
    immi i dati dell'Apple."
620 PRINT : PRINT "POKE :";: INPUT
    PA
630 IF PA < 1024 OR PA > 2047 THEN
    620
640 P1 =  INT ((PA - 1024) / 128)

650 P2 = (PA - 1024) - (128 * P1)

651 P2 = P2 + 1
660 IF P2 > 120 THEN 900
670 IF P2 = > 1 AND P2 = < 40 THEN
    P3 = 1
680 IF P2 = > 41 AND P2 = < 80
    THEN P3 = 2:P2 = P2 - 40
690 IF P2 = > 81 AND P2 = < 12
    0 THEN P3 = 3:P2 = P2 - 80
700 X = P1 + (P3 - 1) * 8 + 1:Y =
    P2
710 PO = 1024 + (Y - 1) + 40 * (X
    - 1)
720 CO = 55296 + (Y - 1) + 40 * (
    X - 1)
```

Seguito listato 1.

```
730 PRINT : PRINT : PRINT "A POK
    E "PA" DELL'APPLE CORRISPOND
    ONO :"
740 PRINT "VTAB"X"     HTAB"Y
750 PRINT : PRINT "E NEL C 64 CO
    RRISPONDONO A :"
760 PRINT "- SCHERMO POKE "PO
770 PRINT "- COLORE  POKE "CO
780 PRINT : PRINT : PRINT "ANCOR
    A ? (S/N) ";
790 GET A$: IF A$ = "N" THEN  END

800 IF A$ = "S" THEN 100
810 GOTO 790
900 PRINT : PRINT : FLASH : PRINT
    "POKE FUORI SCHERMO !!!!"
910 GOTO 780
```

Listato 2. *Il gioco della Ragnatela. Questa è la versione per Apple che può essere facilmente convertita per il C 64 utilizzando il programma del listato 1.*

```
1  GOSUB 3000
2  DIM CA(30)
4 PU = 0:PP = 999:GI = 0:TR = 0:T
    E = 0:RT = 0
5  HOME :L$ = " ":TT = 0
10  PRINT " #####################
    #################"
12  PRINT " #           ###      ##
    ###        #      ##"
14  PRINT " # ########     # ###
    ##### # ##    #"
16  PRINT " # #       #   #  # ### #
    ## ######     # #"
18  PRINT " # # ### # ##          #
    #       # # # # # #"
20  PRINT " # # #     # #   # ### #
    ### # # # # # # #"
22  PRINT " #     # # # # ## #    #
    # #      # # #    #"
24  PRINT " # #           #       #
    #    # #    # ## ##"
26  PRINT " # ###### ## #### # #
    # #      # #   # ##"
28  PRINT " #        #   #     #   # #
    ## ## # ##      #"
30  PRINT " ## ## # ## ##     ##
    #       # ### # #"
32  PRINT " #   ##     #     # # ## #
    ####### #       # #"
34  PRINT " # ####### # # #       #
    # # # #"
36  PRINT " # ###       #    ## # ##
    #### ###### # # #"
38  PRINT " #       ##### # ##
    #### # # #"
40  PRINT " ### #       # # #######
    # ####          #"
```



Schermo Apple - C 64 e la ragnatela

Seguito listato 2.

```
42  PRINT " ### # ### # #
          ##### # ##"
44  PRINT " #    # ###    # ##### #
    ######     # #  #"
46  PRINT " # ###  #### # ###
          #### #  # #"
48  PRINT " # #### ####     ## ###
    #####     # ## # #"
50  PRINT " #                ##
          ##         #"
52  PRINT " ######################
    ##################"
55  PRINT "Punti:";: HTAB (12 -  LEN
     ( STR$ (PU)): PRINT PU;: HTAB
     (22 -  LEN ( STR$ (P))): PRINT
     P;: HTAB 28: PRINT "Record:"
     ;: HTAB (40 -  LEN ( STR$ (R
     E))): PRINT RE
60 X = 2:Y = 3:A = 21:B = 38
70  GOSUB 800
80  GOSUB 600
90  FOR T = 1 TO 1000: NEXT
```

Seguito listato 2.

```
100  VTAB 23: PRINT "Punti:";: HTAB
     (12 -  LEN ( STR$ (PU))): PRINT
     PU;
101  VTAB 23
102  HTAB (21 -  LEN ( STR$ (P)))
     : PRINT " "P;
104  HTAB 28: PRINT "Record:";: HTAB
     (40 -  LEN ( STR$ (RE))): PRINT
     RE
106  VTAB 24: PRINT "Giro : "(GI +
     1);: HTAB 12: PRINT "$="(TE -
     TT);: HTAB 18: PRINT "@= -"(
     4 ^ (RT + 1));
107  HTAB 29: PRINT "Bonus:";: HTAB
     (40 -  LEN ( STR$ (P * (GI +
     1)))): PRINT (P * (GI + 1));
109  VTAB X: HTAB Y: PRINT " ": VTAB
     A: HTAB B: PRINT L$
110 K =  PEEK ( - 16384)
111 A$ =  CHR$ (K)
120  IF  ASC (A$) = 214 THEN Y =
     Y + 1
125  IF  ASC (A$) = 195 THEN Y =
     Y - 1
130  IF  ASC (A$) = 216 THEN X =
     X + 1
135  IF  ASC (A$) = 211 THEN X =
     X - 1
140 H =  INT ( RND (1) * 2)
150  IF H = 0 THEN A = A + (X > A
     ) - (X < A)
160  IF H = 1 THEN B = B + (Y > B
     ) - (Y < B)
165  GOTO 1000
170  VTAB A: HTAB B: PRINT "X"
180  IF A = X AND B = Y THEN 300
181 NN = 163: GOSUB 500
182 Q1 = Q3 + B * (XX - 1)
183 Q2 = Q1 + Y + 40 * ((XX - 1) *
     3)
184  IF  PEEK (Q2) = NN AND  ASC
     (A$) = 214 THEN Y = Y - 1
185  IF  PEEK (Q2) = NN AND  ASC
     (A$) = 195 THEN Y = Y + 1
186  IF  PEEK (Q2) = NN AND  ASC
     (A$) = 216 THEN X = X - 1
187  IF  PEEK (Q2) = NN AND  ASC
     (A$) = 211 THEN X = X + 1
188  IF  PEEK (Q2) = 164 THEN PU =
     PU + 50:TT = TT + 1: IF TT =
     TE THEN 400
189  IF  PEEK (Q2) = 192 THEN RT =
     RT + 1:PU = PU - 4 ^ RT
190  VTAB X: HTAB Y: PRINT "*"
200 P = P - 1: IF P = 0 THEN 310
290  GOTO 100
300  VTAB A: HTAB B: FLASH : PRINT
     "GNAM!!": NORMAL : GOTO 320
310  VTAB 12: HTAB 10: FLASH : PRINT
     "E' SCADUTO IL TEMPO": NORMAL

320  IF PU > RE THEN RE = PU
350  VTAB 24: PRINT "Vuoi giocare
     un'altra partita ? (Y/N) ";
```

Schermo Apple - C 64 e la ragnatela

Seguito listato 2.

```
360  GET K$: IF K$ = "Y" THEN 4
370  IF K$ = "N" THEN  SPEED= 255
     : END
380  GOTO 360
400 GI = GI + 1
410 PU = PU + (P * GI)
420  VTAB 23: FLASH : PRINT "BENE
     !!! HAI TERMINATO UN PERCORS
     O, MA   ECCONE UN ALTRO PIU'
      IMPEGNATIVO........": NORMAL

430  FOR T = 1 TO 4000: NEXT
440  GOTO 5
500  IF X =  > 1 AND X =  < 8 THEN
     Q3 = 1023:XX = X
510  IF X =  > 9 AND X =  < 16 THEN
     Q3 = 1063:XX = X - 8
520  IF X =  > 17 AND X =  < 24 THEN
     Q3 = 1103:XX = X - 16
530  RETURN
600  IF TE < 30 THEN TE = TE + 3
610  IF PP > 400 THEN PP = PP - 1
     00:P = PP
615  RESTORE
620  FOR T = 1 TO TE
630  RESTORE
640  READ A$: IF A$ = "$" THEN 66
     0
650  GOTO 640
660 CA(T) =  INT ( RND (1) * 30) +
     1
670  IF T = 1 THEN 710
680  FOR T1 = 1 TO T - 1
690  IF CA(T) = CA(T1) THEN 660
700  NEXT
710  FOR N = 1 TO CA(T): READ Z: NEXT
720  POKE Z,164
730  NEXT
740  RETURN
800  IF TR < 10 THEN TR = TR + 1
810  FOR T = 1 TO TR
820  RESTORE
830  READ A$: IF A$ = "@" THEN 85
     0
840  GOTO 830
850 CA(T) =  INT ( RND (1) * 10) +
     1
860  IF T = 1 THEN 900
870  FOR T1 = 1 TO T - 1
880  IF CA(T) = CA(T1) THEN 850
890  NEXT
900  FOR N = 1 TO CA(T): READ AA:
      NEXT
910  POKE AA,192
920  NEXT
930  RETURN
1000  IF A >  = 1 AND A <  = 8 THEN
     W3 = 1023:AA = A
1010  IF A =  > 9 AND A <  = 16 THEN
     W3 = 1063:AA = A - 8
1020  IF A =  > 17 AND A <  = 24 THEN
     W3 = 1103:AA = A - 16
1030 W1 = W3 + 8 * (AA - 1)
```

Seguito listato 2.

```
1040 W2 = W1 + B + 40 * ((AA - 1)
      * 3)
1050 L$ = " "
1060  IF  PEEK (W2) = 163 THEN L$
      = "#": GOTO 1100
1070  IF  PEEK (W2) = 164 THEN L$
      = "$": GOTO 1100
1080  IF  PEEK (W2) = 192 THEN L$
      = "@": GOTO 1100
1100  GOTO 170
2000  DATA $,1383,1169,1311,1414,
     1432,1546,1553,1687,1798,182
     9
2010  DATA 1397,1934,1947,1957,10
     85,1200,1223,1323,1335,1460
2020  DATA 1639,1600,1741,1849,19
     64,1989,1114,1121,1264,1367
2050  DATA @,1301,1549,1796,1204,
     1227,1593,1838,1132,1500,151
     5
3000  HOME
3010  VTAB 12: HTAB 14: FLASH : PRINT
     "LA RAGNATELA": NORMAL
3020  FOR T = 1 TO 6000: NEXT
3030  HOME
3040  VTAB 3: PRINT "Il gioco con
     siste nel raccogliere i vari
     tesori disseminati lungo il
     percorso     facendo attenzio
     ne alle trappole e senza";
3050  PRINT "farsi prendere dal g
     uardiano dei tesori."
3055  PRINT "Il tutto nel tempo c
     oncesso."
3060  PRINT : PRINT : INVERSE : PRINT
     "SIMBOLI-SIGNIFICATO-CONSEGU
     ENZE": NORMAL
3070  PRINT : PRINT "  $  tesori
        50 punti"
3080  PRINT "  @  trappole  -4^N.
     trapp.scattate"
3090  PRINT "  *  TU"
3100  PRINT "  X  Guardiano  Se t
     i prende finisce la
             partita."
3110  INVERSE : PRINT "PER MUOVER
     TI PREMI :": NORMAL
3120  PRINT : PRINT "  S  Su"
3130  PRINT "  X  Giu"
3140  PRINT "  C  Sinistra"
3150  PRINT "  V  Destra"
3160  PRINT : PRINT : PRINT "PER
     CONTINUARE PREMI  <RETURN> "
     ;
3170  INPUT A$
3180  RETURN
```

Il C 64 in italiano

Sfruttiamo in modo originale alcune caratteristiche del Commodore

di *Alessandro Guida*

Questo articolo e i programmi in esso contenuti, permettono l'implementazione del BASIC in *italiano*. Quindi, niente più GOTO, READ, ecc., ma i più familiari VA A, LEGGI, e così via. Ma, anche se si conosce alla perfezione l'inglese, questo articolo resta una buona occasione per vedere come il computer gestisce i vari messaggi e le parole chiave, e le possibilità offerte dai banchi di memoria selezionabili del C 64.

Inglese	Italiano	Inglese	Italiano
1 END	*FINE*	2 FOR	*PER*
3 NEXT	*INCREMENTA*	4 DATA	*DATI*
5 INPUT #	*IN #*	6 INPUT	*RICHIEDI*
7 DIM	*DIM*	8 READ	*LEGGI*
9 LET	*SIA*	10 GOTO	*VA A*
11 RUN	*ESEGUI*	12 IF	*SE*
13 RESTORE	*RILEGGI*	14 GOSUB	*CHIAMA*
15 RETURN	*RITORNA*	16 REM	,
17 STOP	*ALT*	18 ON	*SECONDO*
19 WAIT	.	20 LOAD	*CARICA*
21 SAVE	*REG*	22 VERIFY	*VER*
23 DEF	*DEF*	24 POKE	*PONI*
25 PRINT #	*P #*	26 PRINT	*SCRIVI*
27 CONT	.	28 LIST	*LISTA*
29 CLR	*PUL*	30 CMD	*COM*
31 SYS	*LM*	32 OPEN	*APRI*
33 CLOSE	*CHIUDI*	34 GET	*PRENDI*
35 NEW	*CANC*	36 TAB(	*TAB(*
37 TO	*FINO*	38 FN	*FN*
39 SPC(	*SPZ(*	40 THEN	*ALLORA*
41 NOT	.	42 STEP	*PASSO*
43 +	+	44 −	−
45 ★	★	46 /	/
47 ↑	↑	48 AND	*E #*
49 OR	*Ø #*	50 >	>
51 =	=	52 <	<
53 SGN	.	54 INT	*INT*
55 ABS	*ASS*	56 USR	.
57 FRE	*MEM*	58 POS	*PC*
59 SQR	*RAD*	60 RND	*CAS*
61 LOG	*LOG*	62 EXP	.
63 COS	*COS*	64 SIN	*SEN*
65 TAN	*TAN*	66 ATN	*AT*
67 PEEK	*CONT*	68 LEN	*LUN*
69 STR$	*STRINGA*	70 VAL	*VAL*
71 ASC	*CODICE*	72 CHR$	*CAR*
73 LEFT$	*S$*	74 RIGHT $	*D$*
75 MID$	*M $*	76 GO	

Tabella 1. *I comandi BASIC.*

Modifiche al S.O. e all'interprete BASIC

Per poter implementare i comandi e i vari messaggi in italiano sarà necessario apportare delle modifiche sia al sistema operativo che all'interprete BASIC. Su altri computer ciò comporta la completa riscrittura del BASIC e del S.O., per il 64, invece, questo non accade.
Possiamo, infatti, sfruttare la possibilità di escludere la ROM nella quale risiede il firmware e selezionare al suo posto della RAM. Per saperne di più su questa caratteristica del 64 leggete la rubrica "I segreti del C 64" su questo stesso numero.

Quello che faremo sarà, perciò, di ricopiare l'interprete e il sistema operativo dalla ROM alla RAM, e quindi vi apporteremo le modifiche richieste.
Per comodità, divideremo l'argomento in tre parti: 1 - I comandi BASIC; 2 - Gli errori; 3 - I messaggi del sistema operativo.

I primi due punti interessano l'interprete BASIC, il terzo le routine di Kernal (S.O.).

I comandi BASIC

I comandi BASIC sono raggruppati in una tabella che parte dall'indirizzo $A09E (41108) ed occupa esattamente 256 byte.
In questa tabella sono riportati tutti i comandi, sotto forma di caratteri ASCII. Per dividere una parola dal-

Il C 64 in italiano

l'altra, l'ultimo carattere è aumentato di 128 (ha il bit 7 alto).

Quando viene digitata da tastiera una linea o un comando l'interprete BASIC consulta questa tabella con metodo sequenziale. Legge, cioè, tutte le parole, in essa contenute, fino a trovare quella interessata. Nell'eseguire questa operazione viene tenuto il conto delle parole lette, in modo da avere il numero d'ordine del comando. Se la parola non viene trovata in elenco si ha SYNTAX ERROR.

Il numero d'ordine viene usato poi come indice per accedere ad un'altra tabella che contiene gli indirizzi delle routine che eseguono i rispettivi comandi.

Questo porta ad alcune limitazioni nella modifica della tabella delle parole chiavi:

1) Le parole che sostituiremo devono mantenere lo stesso ordine di quelle in inglese.

2) Bisogna che tutte le parole siano presenti.

In tabella 1 sono riportate tutte le parole chiave del BASIC e la loro traduzione. È possibile usare termini diversi da quelli raffigurati, ma è necessario tenere presenti alcune altre restrizioni.

La più pesante è quella che ci obbliga a contenere tutte le parole chiave nei 256 byte occupati dalle originali. Questo perché l'accesso alla tabella viene effettuato dal microprocessore per mezzo di un registro indice da 8 bit, capace di indirizzare, quindi, al massimo 256 locazioni.

Poiché, però, l'italiano non è sintetico come l'inglese è stato necessario sacrificare alcuni comandi raramente utilizzati (CONT, WAIT, NOT, SGN, USR e EXP), ed abbreviarne altri, come PEEK tradotto CONTENUTO e abbreviato in CONT.

```
2 REM************************************
3 REM*                                  *
4 REM*             <BASIT>              *
5 REM*                                  *
6 REM*    KEYWORD BASIC IN ITALIANO     *
7 REM*                                  *
8 REM************************************
9 REM------COPIA INTERPRETE BASIC------
10 FORI=40960TO49151:POKEI,PEEK(I):NEXT
12 REM--------------------------------
15 S=41117:N=0
20 READA$:IFA$="***"THEN1100
30 N=N+LEN(A$):IFN>255THENPRINT"KEYWORD TROPPO LUNGHE":END
40 IFLEN(A$)=1THENI=1:GOTO60
50 FORI=1TOLEN(A$)-1:POKES+I,ASC(MID$(A$,I,1)):NEXT
60 S=S+I
100 POKES,ASC(RIGHT$(A$,1))+128
110 GOTO20
1000 DATAFINE,PER,INCREMENTA,DATI,IN#,RICHIEDI,DIM,LEGGI,SIA,VA A
1010 DATAESEGUI,SE,RILEGGI,CHIAMA,RITORNA,',ALT,SECONDO,£,CARICA
1020 DATAREG,VER,DEF,PONI,P#,SCRIVI,£,LISTA
1030 DATAPUL,COM,LM,APRI,CHIUDI,PRENDI,CANC,TAB(,FINO,FN,SPZ(
1040 DATAALLORA,£,PASSO,+,-,*,/,↑,E#,O#,>,=,<,£,INT,ASS,£
1050 DATAMEM,PC,RAD,CAS,LOG,£,COS,SEN,TAN,AT
1060 DATACONT,LUN,STRINGA,VAL,CODICE,CAR,S$,D$,M$,***
1100 POKES+1,0:REM FINE TABELLA
1110 POKE1,54: REM SELEZIONA RAM
```

Listato 1. *Programma per attivare il BASIC con comandi in italiano.*

```
10 SCRIVI "□■"TAB(18)"BASIT■"
15 SCRIVI CAR(142)
20 PER I=1 FINO 6
30 LEGGI A$
40 PER J=1 FINO LUN(A$)
50 SCRIVI M$(A$,J,1);
55 PER D=0 FINO 200:INCREMENTA D
60 INCREMENTA J
70 SCRIVI:INCREMENTA I
80 LISTA
90 FINE
100 DATI "QUESTO PROGRAMMA IMPLEMENTA SUL TUO"
110 DATI "C64 IL BASIC IN ITALIANO!"
120 DATI"■RICORDA CHE:"
130 DATI"■STOP+RESTORE DISATTIVA BASIT"
140 DATI"■POKE1,54  ATTIVA BASIT"
150 DATI"■■BUON LAVORO !!!!"
```

Listato 1a. *Esempio di programma scritto in BASIT.*

Alle parole soppresse è stato assegnato il carattere "&", perché, come abbiamo visto, è necessario mantenere il posto occupato nella tabella.

Le parole AND e OR che andavano tradotte con E e O hanno subito l'aggiunta del carattere "#", perché, diversamente, avrebbero inibito l'uso di qualsiasi variabile contenente uno di questi due caratteri.

È, infatti, sempre valida la regola che il nome dato alle variabili, in un programma, non deve contenere alcuna parola chiave. Cioè, così come con il BASIC in inglese è proibito usare, ad esempio, la variabile *LET*TO così nel nuovo BASIC non è possibile usare A*SIA* o *PER*SO.

Si può notare che l'elenco delle parole chiave inglesi termina con GO che in effetti non viene utilizzata. Si possono, perciò, utilizzare i suoi due byte, perché essendo l'ultima della serie non pregiudica il calcolo del numero d'ordine degli altri comandi.

Il programma del listato 1 "BASIT", implementa il BASIC in italiano sul vostro computer.

È possibile in ogni momento tornare al BASIC tradizionale premendo STOP + RESTORE. Per riabilitare il BASIT dare POKE1, 54.

Il listato 1a riporta un esempio di programma scritto in BASIT.

ONE HUNDRED STEPS TO LONDON

CONCORSO per tutti i SINCLAIRisti





COME FUNZIONA IL CONCORSO?

Prima di entrare nel dettaglio del Regolamento che per altro consigliamo di leggere attentamente, descriviamo il meccanismo di questo strabiliante concorso, organizzato dalla J.C.E.
Ci preme innanzitutto chiarire che la Giuria è il pubblico, ossia i lettori di Sperimentare con l'Elettronica e il Computer, e ciò sottolinea la filosofia di dialogo e di fiducia, in cui la redazione non si pone, come in altri casi spesso avviene, nella presuntuosa posizione di infallibilità!
Ogni mese a partire dal numero di Giugno 1984, la rivista pubblicherà i quattro migliori programmi pervenuti in redazione, e giudicati dalla nostra commissione tecnica. I lettori troveranno quindi i listati di questi quattro programmi ed un tagliando sul quale scriveranno il titolo del programma che sarà parso più meritevole, per qualsiasi motivo.
Fra tutti i lettori che avranno inviato il tagliando sarà estratto, ogni mese, un computer Sinclair Spectrum 16 K!!!
Questa procedura si ripeterà per sei mesi, quindi sui numeri di Giugno, Luglio/Agosto, Settembre, Ottobre, Novembre e Dicembre, quindi ogni mese saranno pubblicati quattro programmi, il tagliando da spedire e fra i tagliandi sarà estratto uno Spectrum 16 K.
Alla fine di questa prima fase usciranno, dunque sei programmi classificati al primo posto in ciascuna delle sei "batterie".
Noi li sottoporremo al giudizio dei lettori, pubblicando sul numero di Marzo 1985 un nuovo tagliando sul quale andranno indicati, in ordine di preferenza, tutti i sei programmi. I primi tre saranno premiati, e fra i lettori sarà estratto di nuovo uno Spectrum 16 K.
Mica male, eh, che ne dite?
Il regolamento è riportato sulla rivista "Sperimentare con l'Elettronica e il Computer".

1° CONCORSO SINCLUB 1984-85

I Sinclair Club e i Sinclairisti sono spesso vere e proprie fonti di idee per il miglior impiego del nostro beneamato spectrum; è un peccato che la maggior parte di queste idee restino nel cassetto, o nel migliore dei casi vengano resi noti alla ristretta cerchia degli amici.
Uno degli scopi del Sinclub è proprio quello di dare le ali a chi le merita. Da qui l'idea di un concorso a premi, organizzato dalla **J.C.E.**, aperto a tutti. Il concorso inizierà il **1° Aprile** e terminerà Il **31 Dicembre 1984**; tutti potranno inviare alla redazione di **Sperimentare con l'Elettronica e il Computer** entro il suddetto periodo loro programmi più interessanti ed originali.

10) I programmi devono essere memorizzati su cassetta e devono essere spediti alla redazione di Sperimentare con l'Elettronica e il Computer al seguente indirizzo:

J.C.E. - CONCORSO SINCLUB
VIA DEI LAVORATORI, 124
20092 CINISELLO BALSAMO (MI)

PERSONAL SOFTWARE

Il C 64 in italiano

I messaggi di errore in BASIC

La gestione dei messaggi di errore e non, da parte dell'interprete è leggermente più complessa. Esistono, infatti, due tabelle di cui una contiene tutti i messaggi di errore e l'altra gli indirizzi di partenza di tali messaggi. La prima è posta in memoria a partire dalla locazione $A19E fino alla $A327 e contiene i messaggi, riportati in tabella 2, da TOO MANY FILES a LOAD. La seconda, contenente gli indirizzi di inizio, incluso quello di BREAK, parte da $A328 e termina in $A363. La routine di gestione errori, il cui indirizzo di partenza è contenuto nel vettore $0300, accede proprio a quest'ultima tabella per mezzo del registro X del 6502 usato come indice.
Perciò i codici degli errori, da 0 a 29, devono mantenere lo stesso ordine originale, mentre il testo può essere messo in un punto qualsiasi della memoria purché si sia aggiornato il corrispondente elemento della tabella indirizzi.
I vari messaggi sono separati con lo stesso sistema visto per le parole chiavi.
Sfruttando questa possibilità e tenendo presente che concentrare il testo italiano nello stesso spazio occupato dall'inglese era impossibile, si è preferito modificare la tabella codice-indirizzo e memorizzare i testi a partire da $C000 (49152). Con il programma del listato 2 potrete modificare a piacere i testi degli errori poiché il programma provvede a calcolare la lunghezza delle varie

	Inglese	Italiano
1	TOO MANY FILES	*ERRORE: TROPPI FILE*
2	FILE OPEN	*ERRORE: FILE GIA' APERTO*
3	FILE NOT OPEN	*ERRORE: FILE CHIUSO*
4	FILE NOT FOUND	*ERRORE: NON TROVO IL FILE*
5	DEVICE NOT PRESENT	*ERRORE: PERIFERICA ASSENTE*
6	NOT INPUT FILE	*ERRORE: FILE DI USCITA DATI*
7	NOT OUTPUT FILE	*ERRORE: FILE DI INGRESSO DATI*
8	MISSING FILE NAME	*ERRORE: FILE SENZA NOME*
9	ILLEGAL DEVICE NUMBER	*ERRORE: NUMERO PERIFERICA ILLEGALE*
10	NEXT WITHOUT FOR	*ERRORE: INCREMENTO SENZA CICLO*
11	SYNTAX	*ERRORE DI SINTASSI*
12	RETURN WITHOUT GOSUB	*ERRORE: RITORNO SENZA CHIAMATA*
13	OUT OF DATA	*ERRORE: TERMINATI I DATI*
14	ILLEGAL QUANTITY	*ERRORE PER QUANTITA' ILLEGALE*
15	OVERFLOW	*ERRORE: RISULTATO TROPPO GRANDE*
16	OUT OF MEMORY	*ERRORE: MEMORIA ESAURITA*
17	UNDEF'D STATMENT	*ERRORE: NUMERO DI LINEA SCONOSCIUTO*
18	BAD SUBSCRIPT	*ERRORE: VETTORE TROPPO PICCOLO*
19	REDIM'D ARRAY	*ERRORE: VETTORE GIA' DEFINITO*
20	DIVISIO BY ZERO	*ERRORE: DIVISIONE PER ZERO*
21	ILLEGAL DIRECT	*ERRORE: ISTRUZIONE NON DIRETTA*
22	TYPE MISMATCH	*ERRORE: VARIABILE DI TIPO ERRATO*
23	STRING TOO LONG	*ERRORE: STRINGA TROPPO LUNGA*
24	FILE DATA	*ERRORE NEI DATI DI UN FILE*
25	FORMULA TOO COMPLEX	*ERRORE: FORMULA TROPPO COMPLESSA*
26	CAN'T CONTINUE	*MI SPIACE, MA NON POSSO CONTINUARE*
27	UNDEF'D FUNCTION	*ERRORE: FUNZIONE NON DEFINITA*
28	VERIFY	*ERRORE NELLA VERIFICA*
29	LOAD	*ERRORE IN LETTURA*
30	BREAK	*SONO STATO INTERROTTO*
	OK (CR)	*OK (CR)*
	ERROR	
	IN	*IN*
	(CR) (LF) READY (CR) (LF)	*(CR) PRONTO (CR) (LF)*
	(CR) (LF) BREAK	*(CR) (LF) ALT*
	? EXTRA IGNORED (CR)	*? TROPPI DATI*
	? REDO FROM START	*NON VA BENE!*

Tabella 2. *I messaggi BASIC. Nota: (CR) = Carriage return; (LF) = Line Feed.*

frasi e ad aggiornare di conseguenza la tabella indirizzi.

Ma vediamo un attimo cosa accade al verificarsi di un errore:
1) l'interprete BASIC mette nel registro X del microprocessore il codice dell'errore, e salta alla routine gestione errori;
2) utilizzando il registro X come indice viene letto l'indirizzo di partenza del messaggio selezionato.
Questo viene quindi stampato un carattere per volta finché non viene incontrato l'ultimo:
3) viene stampata la parola "ERROR" prelevata in un'altra piccola tabella;
4) se il computer si trova all'interno di un programma stampa "IN" seguito dal numero di linea. Altrimenti salta quest'ultima fase.

Naturalmente, pur traducendo tutto in italiano sarebbe ugualmente brutto leggere, ad esempio: "STRINGA TROPPO LUNGA ERRORE".

Perciò i vari messaggi sono stati tradotti in maniera completa (vedi tabella 2) comprendendo anche la parola ERRORE che è stata invece cancellata nella versione inglese in modo che non venisse ripetuta al termine della frase.

La parola "ERROR" è contenuta in una terza tabella che comprende, anche, i messaggi da OK a BREAK, ed è posta in memoria da $A364 a $A389. L'accesso a questi messaggi avviene in modo diretto, ossia il computer si aspetta sempre di trovarli allo stesso posto per cui è necessario sostituire al testo inglese quello italiano *senza eccedere il numero di caratteri previsto per ognuno dei cinque messaggi.*
Questi testi sono separati uno dall'altro dal carattere $00.
Come già detto "ERROR" è completamente eliminato.

Per terminare, ci sono altri due messaggi a partire da $ACFC relativi all'istruzione di INPUT. Anche questi devono essere rimpiazzati "in situ" come gli ultimi messaggi visti.
Il programma del listato 2 può essere utilizzato da solo o insieme al programma 1. In questo secondo caso le linee 1980 e 4050 devono essere omesse.

```
1900 REM*********************************
1910 REM*                               *
1920 REM*          <ERRIT>              *
1930 REM*                               *
1940 REM* MESSAGGI ERRORE IN ITALIANO *
1950 REM*                               *
1960 REM*********************************
1970 REM------COPIA INTERPRETE BASIC----
1980 FORI=40960TO49151:POKEI,PEEK(I):NEXT
1990 REM----------------------------
1995 REM------COPIA MESSAGGI ERRORE----
2000 TB=41763:IN=49152
2010 READA$:IFA$="***"THEN3500
2020 HB=INT(IN/256):LB=IN-HB*256:POKETB,LB:POKETB+1,HB
2030 FORI=1TOLEN(A$)-1:POKEIN-1+I,ASC(MID$(A$,I,1)):NEXT
2040 POKEIN+I-1,ASC(RIGHT$(A$,1))+128
2050 IN=IN+LEN(A$):TB=TB+2:GOTO2010
3000 DATA"ERRORE: TROPPI FILE","ERRORE: FILE GIA' APERTO"
3010 DATA"ERRORE: FILE CHIUSO","NON TROVO IL FILE"
3020 DATA"PERIFERICA ASSENTE","ERRORE: FILE DI USCITA DATI"
3030 DATA"ERRORE: FILE DI INGRESSO DATI","ERRORE: FILE SENZA NOME"
3040 DATA"ERRORE: NUMERO PERIFERICA ILLEGALE","ERRORE: INCREMENTO SENZA PER"
3050 DATA"ERRORE DI SINTASSI","ERRORE: RITORNO SENZA CHIAMATA"
3060 DATA"ERRORE: TERMINATI I DATI","ERRORE PER QUANTITA' ILLEGALE"
3070 DATA"ERRORE: RISULTATO TROPPO GRANDE","MEMORIA ESAURITA"
3080 DATA"ERRORE: NUMERO DI LINEA SCONOSCIUTO","ERRORE: VETTORE NON DEFINITO"
3090 DATA"ERRORE: VETTORE GIA' DEFINITO","ERRORE: DIVISIONE PER ZERO"
3100 DATA"ERRORE: ISTRUZIONE NON DIRETTA","ERRORE: VARIABILE DI TIPO ERRATO"
3110 DATA"ERRORE: STRINGA TROPPO LUNGA","ERRORE NEI DATI DI UN FILE"
3120 DATA"ERRORE: FORMULA TROPPO COMPLESSA","MI SPIACE, MA NON POSSO CONTINUARE"
3130 DATA"ERRORE: FUNZIONE NON DEFINITA","ERRORE NELLA VERIFICA"
3140 DATA"ERRORE IN LETTURA","SONO STATO INTERROTTO",***
3490 REM-------CANCELLA 'ERROR'-------
3500 POKE41834,0
3505 REM---RIMPIAZZA ALTRI MESSAGGI---
3510 FORI=1TO6:POKE41847+I,ASC(MID$("PRONTO",I,1)):NEXT
3520 FORI=1TO5:POKE41858+I,ASC(MID$("ALT  ",I,1)):NEXT
4000 READA$:FORI=1TOLEN(A$):POKE44283+I,ASC(MID$(A$,I,1)):NEXT
4010 POKE44283+I,13:POKE44283+I+1,0
4020 READA$:FORI=1TOLEN(A$):POKE44299+I,ASC(MID$(A$,I,1)):NEXT
4030 POKE44299+I,13:POKE44299+I+1,0
4040 DATA"? TROPPI DATI","NON VA BENE !"
4050 POKE1,54: REM SELEZIONA RAM
```

Listato 2. *Programma "ERRIT" per implementare i messaggi di errore in italiano sul C 64.*

	Inglese	Italiano
1	I/O ERROR #	*ERRORE I/O #*
2	SEARCHING	*CERCO:*
3	FOR	
4	PRESS PLAY ON TAPE	*PREMERE TASTO PLAY*
5	PRESS RECORD AND PLAY ON TAPE	*PREMERE TASTI RECORD E PLAY*
6	LOADING	*CARICO*
7	SAVING	*SALVO*
8	VERIFYING	*VERIFICO*
9	FOUND	*TROVATO*
10	OK	*OK*

Tabella 3. *Messaggi del sistema operativo.*

I messaggi del sistema operativo

Anche il sistema operativo possiede un certo numero di messaggi, riguardanti le operazioni di input/output, che possono essere tradotti. L'elenco con la traduzione proposta è in tabella 3.
La tabella originale è all'indirizzo $F0BD. Anche in questo caso, l'accesso, da parte del computer ai testi, avviene in modo diretto puntando



Il C 64 in italiano

```
9900 REM********************************
9910 REM*                              *
9920 REM*           <KERNIT>           *
9930 REM*                              *
9940 REM*  MESSAGGI S.O. IN ITALIANO   *
9950 REM*                              *
9960 REM********************************
9970 REM----COPIA INTERPRETE BASIC----
9980 FORI=40960TO49151:POKEI,PEEK(I):NEXT
9990 REM----COPIA SISTEMA OPERATIVO---
10000 FORI=57344TO65535:POKEI,PEEK(I):NEXT
10002 REM---MODIFICA MESSAGGI I/O-----
10005 FORI=1TO9
10010 READA,A$
10020 FORJ=1TOLEN(A$)-1:POKEA+J,ASC(MID$(A$,J,1)):NEXTJ
10030 POKEA+J,ASC(RIGHT$(A$,1))+128:NEXTI
10100 DATA61628,"ERRORE I/O #",61641,"CERCO :",61651,"    "
10110 DATA61656,"PREMERE TASTO PLAY",61674,"PREMERE TASTI RECORD E PLAY"
10120 DATA61702,"CARICO ",61710,"SALVO "
10130 DATA61718,"VERIFICO ",61727,"TROVATO"
10140 POKE1,53: REM SELEZIONA RAM
```

Listato 3. *Programma per attivare i messaggi di I/O in italiano.*

```
PRONTO

CARICA "KERNIT",8

SEARCHING FOR KERNIT
LOADING
PRONTO

ESEGUI
ERRORE DI SINTASSI IN 10010
LISTA 10010

10010 LEGGI A;A$
```

Figura 1. *Esempio di utilizzo dei comandi e dei messaggi di errore in italiano.*
Da notare che i messaggi del sistema operativo sono ancora in inglese.

```
PRONTO

CARICA "ESEMPIO",8

CERCO: ESEMPIO
?NON TROVO IL FILE
PRONTO

CARICA "ESEMPIO"

PREMERE TASTO PLAY
CERCO:  ESEMPIO
CARICO
PRONTO

ESEGUI 1000
```

Figura 2. *Esempio di utilizzo contemporaneo di "BASIT", "ERRIT" e "KERNIT". In questo caso anche i messaggi provenienti dal sistema operativo "Kernal" sono in italiano.*

direttamente al messaggio da stampare. Perciò nella sostituzione è importante che i nuovi messaggi non siano più lunghi degli originali.

Poiché, nel selezionare la parte alta della RAM, ($E000 - $FFFF) corrispondente al sistema operativo, viene automaticamente attivata anche la RAM corrispondente all'interprete BASIC ($A000 - $BFFF), è necessario ricopiare tale zona di memoria prima di attivare la RAM.

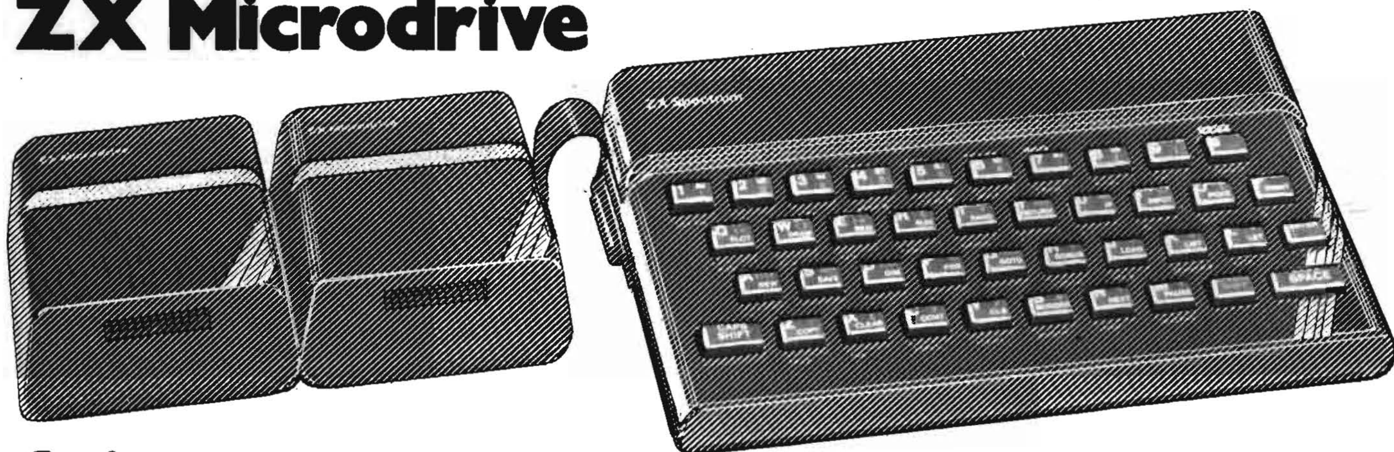
Naturalmente, se il programma "KERNIT" riportato nel listato 3, è utilizzato insieme agli altri visti prima la linea 9980 va tolta.

Le figure 1 e 2 riportano due esempi di utilizzo dei 3 programmi.

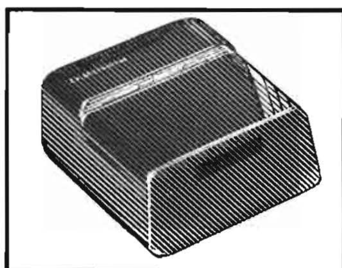
Conclusioni

Questo visto è solo un esempio di quanto è possibile fare sfruttando la selezione dei banchi di memoria da $A000 in su.
In futuro si vedranno altri esempi, tutti molto interessanti, di manipolazione dell'interprete BASIC e del sistema operativo. ■

ZX Microdrive

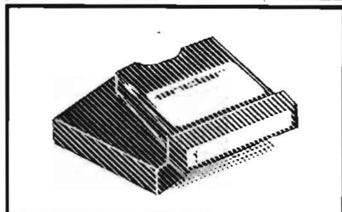


fai crescere il tuo sinclair - Spectrum con le sue eccezionali periferiche!



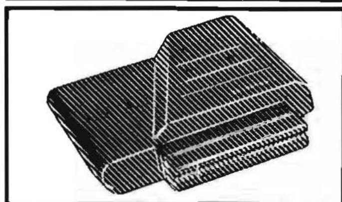
ZX MICRODRIVE
Amplia le possibilità dello ZX Spectrum in quei settori come la didattica e le piccole applicazioni gestionali, dove è necessaria una ricerca veloce delle informazioni.
Ogni cartuccia può contenere: 85 kbyte / 95 kbyte

L.199.500



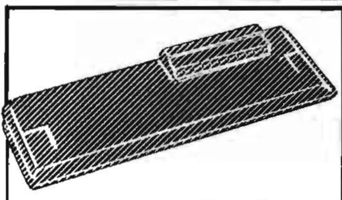
CARTRIDGE
Per ZX Microdrive.
Capacità: 85 kbyte / 95 kbyte
Confezione da 2 pezzi.

La coppia L. 45.000



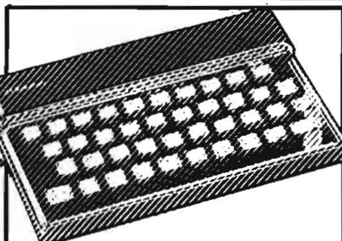
ZX INTERFACE 2
Permette di utilizzare le nuovissime ZX ROM, cartucce software e il collegamento per 2 joystick.

L. 95.500



ZX INTERFACE 1
Indispensabile per il collegamento dello ZX Microdrive.
Inoltre permette il collegamento fra lo ZX Spectrum e una ampia gamma di periferiche e di altri Sinclair in rete locale.

L.199.500



COMPUTER ZX SPECTRUM
A colori, collegabile ad un televisore a colori o in b/n e ad un normale registratore a cassetta.
32x24 caratteri.
RAM di base: 16 k - 48 k
256x192 punti.
8 colori - 2 luminosità.

16 K L.398.000

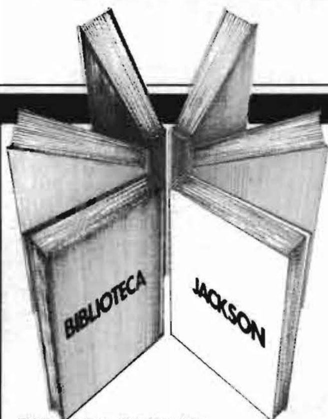
48 k L.499.000

I prezzi sono comprensivi di I.V.A.



competenza in COMPUTER

sinclair



Libri firmati JACKSON



GIOCARE IN BASIC
Il gioco come metodo d'apprendimento del BASIC e dei microcomputer
324 pag. **L. 20.000**
Cod. **522A**

PROGRAMMI SCIENTIFICI IN PASCAL
Per costruirsi una "libreria" di programmi in grado di risolvere i più frequenti problemi scientifici e ingegneristici
384 pag. **L. 25.000**
Cod. **554P**

DAL FORTRAN IV AL FORTRAN 77
Per chi deve programmare a livello tecnico scientifico e per chi vuole approfondire le conoscenze del linguaggio
266 pag. **L. 18.000**
Cod. **517P**

IL BASIC DEL PET E DELL'M20
Un validissimo supporto e strumento di lavoro per chiunque voglia o debba imparare a programmare in BASIC con un Commodore o un Olivetti M20
232 pag. **L. 16.000**
Cod. **336D**

FORTH PER VIC 20 E CBM 64
Il libro spiega la programmazione in Forth: linguaggio che dopo essersi affermato in campo scientifico ed industriale, sta ora diffondendosi anche a livello di personal computer.
156 pag. **L. 11.000**
Cod. **527B**

La Biblioteca che fa testo

IL BASIC PER TUTTI
Per i neofiti una facile e immediata introduzione al linguaggio BASIC e al mondo dei calcolatori
264 pag. **L. 17.500**
Cod. **525A**

50 ESERCIZI IN BASIC
Una raccolta completa e progressiva di esercizi matematici, gestionali, operativi, statistici, di svago
208 pag. **L. 13.000**
Cod. **521A**



GRUPPO EDITORIALE JACKSON

Attenzione compilare per intero la cedola
ritagliare (o fotocopiare) e spedire in busta chiusa a:
GRUPPO EDITORIALE JACKSON
Divisione Libri
Via Rosellini, 12 - 20124 Milano

CEDOLA DI COMMISSIONE LIBRARIA

VOGLIATE SPEDIRMI

n° copie	codice	Prezzo unitario	Prezzo totale
		Totale	

☐ Pagherò contrassegno al postino il prezzo indicato più L. 2000 **per contributo fisso spese di spedizione**

Condizioni di pagamento con esenzione del contributo spese di spedizione:

☐ Allego assegno della Banca ______
n° ______

☐ Allego fotocopia del versamento su c/c n. 11666203 a voi intestato

☐ Allego fotocopia di versamento su vaglia postale a voi intestato

Nome ______
Cognome ______
Via ______
Cap ______ Città ______ Prov. ______
Data ______ Firma ______

Spazio riservato alle Aziende. Si richiede l'emissione di fattura
Partita I.V.A. ☐☐☐☐☐☐☐☐☐☐☐

ORDINE MINIMO L. 50.000

Analisi delle serie temporali

Lo Spectrum disegna gli istogrammi

di *Vittorio Riva*

A causa della turbolenza che ha caratterizzato l'ambiente economico negli ultimi anni, l'analisi dei dati storici è diventata strumento di verifica e di previsione di fondamentale importanza nella gestione aziendale.
Per le aziende di piccole dimensioni il discorso assume un importanza ancora più rilevante poiché queste dispongono generalmente di margini considerevolmente ridotti per far fronte ad eventuali errori derivanti da valutazioni approssimative ed interpretazioni arbitrarie dei dati di mercato.
Fortunatamente il problema può essere risolto con un minimo impiego di capitale, infatti anche sistemi di piccole dimensioni, come lo Spectrum, consentono di effettuare valutazioni matematiche accurate, contribuendo ad eliminare parte della casualità delle scelte.
Quale professionista non si è mai posto il problema di tenere sotto controllo l'andamento (trend), per es., dei propri incassi e dei propri costi?

Ma anche per la vostra gestione domestica può risultare utile avere sotto controllo l'andamento delle spese, quindi il programma presentato, anche se apparentemente sembra destinato solo a persone con un grosso giro di affari, oppure a piccole società, è invece utile a tutti i puntigliosi amanti dei bilanci personali. E se non sapete cosa sia un'istogramma? Beh, niente paura, provate il programma e non ne potrete avere un'idea più chiara.

Il programma

Il programma è stato elaborato sullo Spectrum Sinclair utilizzato in configurazione minima (16 Kbyte ovvero 8,6 Kbyte effettivamente disponibili).
Il calcolatore non si limita al disegno dell'istogramma ma fornisce i dati principali per rendersi conto della situazione passata, presente e, con l'approssimazione consentita, anche della situazione futura. Insomma risponde pienamente alla domanda: "Qual'è la situazione nell'andamento dei dati riguardo ad una determinata classe di valori?".
In Input vengono richiesti:

- titolo dell'analisi (esempio vendite);
- primo periodo (è l'anno zero: da cui si comincia a possedere dati. È il calcolatore stesso che precisa come immettere il dato e ne controlla la correttezza;
- sottoperiodi (t = 1 ... n. Ancora una volta è il calcolatore a precisare la richiesta);
- dati del singolo periodo ($0 < V < 100000$ in qualunque forma e approssimazione. Controllati dal calcolatore).

L'asse delle ascisse dell'istogramma è diviso in quattro periodi. Ogni periodo è formato da 12 sottoperiodi evidenziati a gruppi di tre. È chiaro che la divisione più naturale, a questo punto, è proprio quella più frequente che usa l'anno come periodo e il mese come sottoperiodo (per un totale di 48 mesi). Tuttavia con opportune modifiche e raggruppamenti di dati la scala temporale, di per sé rigida, risulta utilizzabile molto più flessibilmente. La scelta rigida stata imposta dalla ridotta disponibilità di memoria.
In output vengono forniti:

- massimo e minimo (dei dati inseriti);
- valore cumulato ($\sum_{0}^{n} {}_{t}V_{t}$);
- media dati

$$\left(\frac{\Sigma V_t}{\text{n. sottop.}} \right);$$

- S.Q.M. (scarto quadratico medio);
- retta del trend (metodo dei minimi quadrati);
- coeff. di regressione lineare (misura la bontà di un'approssimazione lineare. r = 1 retta r = 0 pessima approssimativamente);
- incrementi mensili % (mese "n" rispetto a mese "n − 1").

Successivamente, una volta disegnato l'istogramma sul cui asse sono riportati i valori approssimati per difetto alla terza cifra decimale è consentito un colloquio interattivo con il calcolatore per richiedere il valore esatto di un dato immesso, nel caso fosse di difficile interpretazione sull'asse delle ordinate. È anche possibile tornare alla stampa dei dati statistici. In più se i dati sono linearizzabili con buona approssimazione ($r > 0.75$), il programma fornisce su richiesta le proiezioni per i tre mesi successivi all'ultimo mese in cui si dispone dei dati. La proiezione non può essere la previsione esatta del futuro ma è tanto più vicina al vero quanto più l'andamento dei dati è approssimabile con una retta e non ci sono brusche variazioni di mercato. Per questo sarebbe inutile andare oltre i tre mesi con una simile proiezione.

Ma è probabile che sia già molto in considerazione del fatto che il nostro obiettivo non è prevedere il futuro, ma di valutare l'andamento dei dati in nostro possesso. Esaminando il trend dei dati è possibile individuare le modifiche da inserire nelle politiche e strategie, per adeguarle agli obiettivi prestabiliti. Il

tempo massimo di elaborazione per 48 dati è di circa 10 sec.. Il programma è protetto contro possibili (ma non clamorosi) errori di battitura nella fase di input, infatti si ferma e attende la correzione oppure risponde coerentemente ogni qualvolta il dato inserito non è compatibile con l'input richiesto.

Alla linea 1671 viene richiesto, prima di uscire definitivamente dal programma, se si desidera copia su cassetta dell'istogramma disegnato. In caso di risposta affermativa viene eseguita copia del video alla linea 3005 attraverso l'istruzione SAVE "*nome*" SCREEN$. L'utente può stabilire con quale nome archiviare il proprio istogramma.

L'immagine così memorizzata può essere richiamata in qualunque momento con l'istruzione LOAD "nome" SCREEN$.

L'istruzione SCREEN$ equivale alla CODE 16384, 6912 ed è in grado di ridisegnare sul video l'istogramma preregistrato.

Questo meccanismo consente di creare un archivio utilissimo per qualunque tipo di confronto con situazioni future. Per valutare il presente è di fondamentale importanza mantenere buona memoria del passato!

Il programma si articola in 6 sezioni principali:

- input (passi 20 ÷ 700);
- dati statistici (passi 940 ÷ 1095);
- disegno assi (passi 1095 ÷ 1281);
- disegno istogrammi (passi 1410 ÷ 1669);
- dialogo interattivo (passi 1669 ÷ 1692);
- retta del trend nella forma $V = a_0 t + a_1$ (passi 1800 ÷ 2034).

Un ultimo avvertimento: il programma occupa 8,3 Kbyte prima del RUN e per poterlo utilizzare con lo Spectrum 16 Kbyte è stata aggiunta l'istruzione "CLEAR 32767" alla linea 10. Con questo accorgimento viene utilizzata anche l'area di memoria riservata ai caratteri grafici definiti dall'utente. Occorre dunque ricordarsi di spegnere il calcolatore dopo aver utilizzato il programma, in modo da ripristinare quest'area per successivi utilizzi grafici.

Inutile dire che lo Spectrum con 48 Kbyte non necessita di questo accorgimento, e quindi la linea 10 può essere cancellata. ■

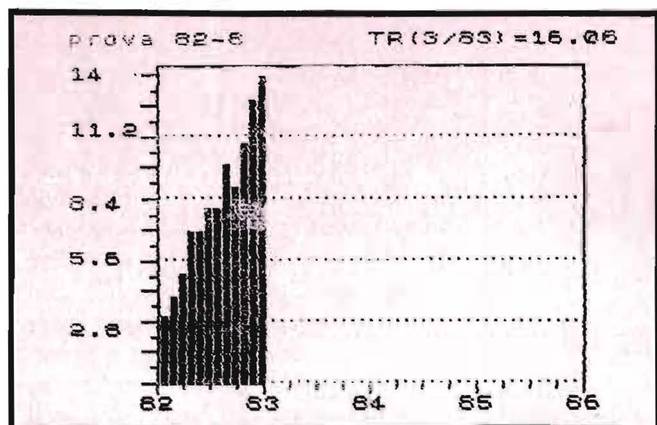


Figura 1. *Istogramma di un periodo (anno 1982) di 12 mesi. I dati introdotti hanno un'ottima correlazione e indicano chiaramente la tendenza (trend) fortemente ascendente. È stato chiesto il valore previsto per il terzo periodo (mese) dell'anno 1983; la previsione è visibile nell'angolo in alto a destra (16.06).*

Figura 2. *Dati statistici relativi all'istogramma della figura 1. Si noti l'elevato coefficiente di correlazione (0.978).*

```
*MAX.= 14
*MIN.= 3
*VAL. CUMULATO= 99
*MEDIA dati= 8.25
*SQM= 3.3876124
```

```
*RETTA TREND
 V=0.91958042*t + 3.1923077

*COEFF. CORREL.= 0.97868299


* PREMI 'CONT' PER CONTINUARE
```

```
2/82 *INCR. sottop.  +33.333333%
3/82 *INCR. sottop.  +25%
4/82 *INCR. sottop.  +40%
5/82 *INCR. sottop.  +0%
6/82 *INCR. sottop.  +14.285714%
7/82 *INCR. sottop.  +0%
8/82 *INCR. sottop.  +25%
9/82 *INCR. sottop.  -10%
10/82 *INCR. sottop.  +22.222222
%
11/82 *INCR. sottop.  +18.181818
%
```

Figura 3. *Primo quadro video di incrementi percentuali di ogni mese rispetto al precedente. I quadri successivi si ottengono rispondendo affermativamente alla domanda "scroll?".*

Analisi delle serie temporali

Listato 1. *Listato BASIC del programma "Istogrammi" per ZX Spectrum. Con Spectrum 48 Kbyte è opportuno eliminare la linea 10.*

```
  10 CLEAR 32767
  20 CLS
  40 BORDER 7: INK 0: PAPER 7: C
LS
  45 DIM A(48): DIM G(48): DIM 3
(48): DIM T(48): DIM P(48): LET
V=0: CLS
 200 PRINT AT 1,0;"***** DISEGNO
 ISTOGRAMMI *****"
 365 PRINT AT 5,0;"INSERISCI IL
TITOLO DELL' ISTOG.(max. 8 lette
re - 0 per uscire)"
 450 INPUT "TITOLO: ";W$
 451 IF W$="0" THEN CLS : STOP
 452 PRINT AT 5,0;"

                    "
 453 IF LEN W$>8 THEN LET W$=W$(
1 TO 8)
 454 PRINT AT 5,0;"** TITOLO= '"
;W$;"'": PAUSE 20
 455 PRINT AT 7,0;"INDICA IL PRI
MO PERIODO           (da 1 a 99: e
x. 83 e non 1983)"
 480 INPUT "PRIMO periodo= ";E
 481 IF E>99 THEN GO TO 480
 485 PRINT AT 7,0;"

                    "
 486 PRINT AT 7,0;"** PRIMO PERI
ODO= ";E
 500 PRINT AT 10,0;"INSERISCI IL
 N. DI SOTTOPERIODI SUCCESSIVI D
I CUI DISPONI"
 501 PRINT AT 13,0;"           **
ex. se PERIODO=anno
 ==> SOTTOPER.=mese"
 510 INPUT "N. TOT. SOTTOPER.(MA
X=48)    ";R
 520 IF R>48 THEN GO TO 510
 539 LET U=R-1
 540 LET T1=1: LET T2=T1: FOR T=
0 TO U
 550 CLS : PRINT : LET Y=0: LET
Y=INT (T/12): LET I=E+Y
 600 PRINT ;T2;"  DATO PER ";I
 610 IF T=12 THEN RESTORE
 620 IF T=24 THEN RESTORE
 630 IF T=36 THEN RESTORE
 640 IF T=48 THEN RESTORE
 650 PRINT
 680 INPUT Q
 681 IF Q>=0 AND Q<100000 THEN G
O TO 690
 682 BEEP .5,3: PRINT AT 20,0;"e
rrore! ridigitare il dato": PAUS
E 60: PRINT AT 20,0;"
                  ": GO TO 680
 690 LET A(T1)=Q: LET B(T1)=Q: L
ET T1=T1+1: LET T2=T2+1
 691 IF T2>12 THEN LET T2=1
 700 LET E1=T1: NEXT T
 940 REM *** TROVA MAX. E MIN.
 950 LET II=I: LET U=A(1): LET W
=999999
 960 FOR X=1 TO R
 970 IF A(X)=0 THEN GO TO 1000
 980 IF A(X)>U THEN LET U=A(X)
 990 IF A(X)<W THEN LET W=A(X)
1000 NEXT X
1010 IF U=0 THEN PRINT "DATI INS
UFFICIENTI!": PAUSE 80: GO TO 45
3
1021 CLS
1022 GO SUB 1730
1030 CLS : PRINT "*MAX.= ";U: PR
INT "*MIN.= ";W;';
1031 LET SSR=0
1032 FOR X=1 TO R: LET SSR=SSR+A
(X): NEXT X: LET SS=SSR/R:
1033 LET MD=0: FOR X=1 TO R: LET
 MD=MD+((A(X)-SS)*(A(X)-SS)): NE
XT X: LET MAD=SQR (MD/(R-1))
1034 PRINT "*VAL. CUMULATO= ";SS
R
1035 PRINT "*MEDIA dati= ";SS
1037 PRINT "*SQM= ";MAD
1038 PRINT "": PRINT "": PRINT "
*RETTA TREND
 V=";AE;"*t + ";AU;';
1039 PRINT "": PRINT "*COEFF. CO
RREL.= ";DD
1040 PRINT AT 21,0;"* PREMI 'CON
T' PER CONTINUARE": PAUSE 0
1041 CLS : LET J=0: LET J1=0
1042 LET MM=2: LET AA=E
1077 FOR X=1 TO (R-1)
1079 LET PU=A(X): LET PD=A(X+1)
1082 LET DDL=PU-PD: IF DDL<0 THE
N GO TO 1084
1083 LET DL=PU-PD: LET C=100*(DL
/PU): GO TO 1085
1084 LET DL=PD-PU: LET C=(DL/PU)
*100
1085 IF DDL>0 THEN GO TO 1088
1086 PRINT MM;"/";AA;" *INCR. so
ttop.  +";C;"%": GO TO 1089
1088 PRINT MM;"/";AA;" *INCR. so
ttop.  -";C;"%"
1090 PRINT "": LET MM=MM+1
1091 IF MM<=12 THEN GO TO 1093
1092 LET AA=AA+1: LET MM=1
1093 NEXT X
1094 PRINT "* PREMI 'CONT' PER C
ONTINUARE"
1095 PAUSE 0
1140 CLS : BORDER 0: PAPER 0: IN
K 5: CLS
1149 BRIGHT 1: LET C1=U: LET C9=
C1: IF C1<=9999 THEN GO TO 1155
1150 LET C1=C1/1000: LET C$=STR$
 C1: IF LEN C$>4 THEN LET C$=C$(
1 TO 4): LET C1=VAL C$: GO TO 11
74
1155 IF C1<=999 THEN GO TO 1160
1156 LET C1=C1/100: LET C$=STR$
C1: IF LEN C$>4 THEN LET C$=C$(1
 TO 4): LET C1=VAL C$: GO TO 117
4
1160 IF C1<=99 THEN GO TO 1164
1161 LET C1=C1/10: LET C$=STR$ C
1: IF LEN C$>4 THEN LET C$=C$(1
TO 4): LET C1=VAL C$: GO TO 1174
1164 IF C1>=10 THEN GO TO 1173
1165 IF C1<=1 THEN GO TO 1172
1166 LET C1=C1*10: LET C$=STR$ C
1: IF LEN C$>4 THEN LET C$=C$(1
TO 4): LET C1=VAL C$
1167 GO TO 1174
1172 LET C$=STR$ C1: IF LEN C$>4
 THEN LET C$=C$(1 TO 4): LET C1=
VAL C$: LET C1=C1*100: GO TO 117
4
1173 LET C$=STR$ C1: IF LEN C$>4
 THEN LET C$=C$(1 TO 4): LET C1=
VAL C$
1174 LET MN=INT C1
1175 LET M3=C1
1180 LET FRMT=(134/U)
1191 PLOT 232,20
1210 FOR A=40 TO 232 STEP 12
1220 PLOT A,20: DRAW 0,-2: NEXT
A: PLOT 40,159: DRAW 192,0: PLOT
 232,20: DRAW 0,139
1230 FOR A=40 TO 232 STEP 48: PL
OT A,20: DRAW 0,-4: NEXT A
1249 PLOT 40,20: DRAW 0,139
1250 FOR A=20 TO 155 STEP 6.75
1260 PLOT 37,A: DRAW 3,0: NEXT A
1270 FOR A=155 TO 20 STEP -13.5
1280 PLOT 33,A: DRAW 5,0: NEXT A
1281 BRIGHT 0: GO SUB 1700
1410 LET Y=1
1420 FOR P=43 TO 232 STEP 4
1430 LET B1=A(Y)*FRMT: IF B1=0 T
HEN GO TO 1491
1450 PLOT P,20+B1: DRAW 0,-B1
1460 PLOT P+1,20+B1: DRAW 0,-B1
1470 PLOT P-1,20+B1: DRAW 0,-B1
1491 LET Y=Y+1: IF Y=R+1 THEN GO
 TO 1500:
1492 NEXT P
1500 PRINT AT 20,4;E;AT 20,10;E+
```

Seguito listato 1.

```
1;AT 20,16;E+2;AT 20,22;E+3;AT 2
0,26;E+4: PRINT AT 0,0;W$;" ";E;
"-";II
1510 LET I=II: IF I=E THEN PRINT
 AT 0,10;"                 "
1550 LET I=2*(M3/10)
1555 LET K=1
1560 FOR Y=I TO (M3+1) STEP I
1570 LET B(K)=Y
1580 LET K=K+1
1590 NEXT Y
1600 LET MN1=INT B(1): LET MN2=I
NT B(2): LET MN3=INT B(3): LET M
N4=INT B(4): LET MN5=INT B(5)
1605 IF B(1)-MN1>0 THEN LET N1=0
: GO TO 1607
1606 LET N1=2
1607 IF B(2)-MN2>0 THEN LET N2=0
: GO TO 1609
1608 LET N2=2
1609 IF B(3)-MN3>0 THEN LET N3=0
: GO TO 1611
1610 LET N3=2
1611 IF B(4)-MN4>0 THEN LET N4=0
: GO TO 1613
1612 LET N4=2
1613 IF B(5)-MN5>0 THEN LET N5=0
: GO TO 1640
1615 LET N5=2
1640 PRINT AT 16,N1;B(1): PRINT
AT 12,N2;B(2): PRINT AT 9,N3;B(3
): PRINT AT 5,N4;B(4)
1650 IF B(5)<100 THEN GO TO 1652
1651 PRINT AT 2,0;B(5): GO TO 16
53
1652 PRINT AT 2,N5;B(5)
1653 IF C9>9999 THEN PRINT AT 0,
17;"(V x1000)"
1654 IF C9>999 AND C9<=9999 THEN
 PRINT AT 0,17;"(V x100)"
1656 IF C9>99 AND C9<=999 THEN P
RINT AT 0,17;"(V x10)"
1657 IF C9<10 AND C9>=1 THEN PRI
NT AT 0,17;"(V:10)"
1658 IF C9<1 THEN PRINT AT 0,17;
"(V :100)"
1669 INPUT ;"CHIEDI sottop. (0 p
er uscire) ";M: IF M<>0 THEN GO
TO 1672
1670 PRINT AT 21,4;"premi 'ENTER
' per uscire": INPUT "'S' per to
rnare a dati stat.";j$: IF j$="S
" OR j$="s" THEN GO TO 1030
1671 INPUT "COPI L'ISTOG. SU FIL
E?(S/N)";R$: GO TO 3000
1672 INPUT ;"              periodo ";
A
1673 IF A<E OR A>=(E+4) THEN GO
TO 1672
1674 IF A=(E+3) THEN LET S=M+36
1675 IF A=(E+2) THEN LET S=M+24
1676 IF A=(E+1) THEN LET S=M+12
1677 IF A=E THEN LET S=M
1678 IF S<=R THEN GO TO 1690
1679 IF S<=(R+3) THEN GO TO 1681
1680 PRINT AT 0,17;"NESSUN VALOR
E": PAUSE 50: GO TO 1692
1681 PRINT AT 0,17;"NESSUN VALOR
E": PAUSE 50
1682 IF DD<.75 THEN GO TO 1692
1683 INPUT "VUOI LA PROIEZIONE D
EL TREND?    (S/N) ";H$
1684 IF H$="N" THEN GO TO 1692
1685 IF H$="S" THEN GO TO 1687
1666 IF H$<>"S" OR H$<>"N" THEN
GO TO 1683
1687 PRINT AT 0,17;"
 ": LET TR=AE*(S-1)+AU: LET TR=T
R*100: LET TR=INT TR: LET TR=TR/
100: PRINT AT 0,17;"TR(";M;"/";A
;")=";TR: PAUSE 220: GO TO 1692
1690: PRINT AT 0,17;"V(";M;"/";A
;")=";A(S): PAUSE 220
1692 PRINT AT 0,17;"
   ": GO TO 1653
1700 FOR A=129 TO 20 STEP -27
1705 FOR Y=40 TO 232 STEP 4
1710 PLOT Y,A: NEXT Y
1711 NEXT A
1720 RETURN
```

Seguito listato 1.

```
1800 CLS
1810 LET X=0
1820 FOR Z=1 TO R-1
1830 LET X=X+Z: NEXT Z
1840 LET MX=X/R
1850 LET X2=0: LET CC=0
1860 FOR X=1 TO R
1870 LET P(X)=(CC-MX)
1880 LET X2=X2+P(X)*P(X): LET CC
=CC+1: NEXT X
1890 LET B=0
1900 FOR Z=1 TO R
1910 LET B=B+A(Z): NEXT Z
1920 LET MY=B/R: LET B2=0
1925 LET B2=0
1930 FOR X=1 TO R
1940 LET T(X)=(A(X)-MY): LET B2=
B2+T(X)*T(X): NEXT X
1950 LET XY=0
1960 FOR X=1 TO R
1970 LET XY=XY+P(X)*T(X): NEXT X
1980 LET AE=XY/X2
1990 LET AU=MY+(AE*(-MX))
2000 LET Z=1: LET D=0
2010 FOR X=1 TO R
2015 LET F=A(X)
2020 LET D=D+(((F-T(Z))*(F-T(Z))
)/R): LET Z=Z+1: NEXT X
2025 LET SYX=SQR D
2027 LET DD=XY/SQR (X2*B2)
2034 RETURN
3000 IF R$="N" OR R$="n" THEN GO
 TO 3010
3003 INPUT "ident. su file= ";a$
3005 SAVE a$ SCREEN$
3010 PAUSE 0: GO TO 10
```

Impariamo il linguaggio macchina con il VIC e il C64

— Parte terza —

Finalmente cominciamo a scrivere i primi programmini

di *Alessandro Guida*

Dove memorizzare i programmi in linguaggio macchina

Prima di iniziare a scrivere dei programmi in linguaggio macchina è giusto chiedersi dove è possibile memorizzarli. Sappiamo, infatti, che parte della memoria del computer è del tipo ROM (memoria a sola scrittura).
Inoltre, esclusa la ROM, la restante memoria è in gran parte utilizzata dal BASIC e dal sistema operativo. In pratica abbiamo solo tre possibilità. Una è di sottrarre un po' di spazio ai programmi BASIC. Per fare ciò, però, è necessario spostare i puntatori che indicano il limite della memoria utilizzabile, altrimenti le variabili usate nei programmi BASIC andrebbero a cancellare le nostre routine in linguaggio macchina.

La seconda possibilità, è di utilizzare il buffer del registratore a cassette, a patto di non effettuare operazioni sul nastro mentre questo è occupato dai programmi in linguaggio macchina.
Questo buffer è lungo 196 byte a partire dalla locazione 828 in decimale ($033C), ed è sufficiente a contenere i primi programmini che scriveremo.
Infine ricordiamo che sul 64 sono disponibili ben 4 Kbyte di RAM per i programmi in linguaggio macchina, a partire da$C000 fino a $CFFF. Invece, per il VIC (con almeno 11 Kbyte di espansione) è disponibile la zona di memoria da $0400 a$0FFF.

Sintassi dei comandi del linguaggio macchina

Per semplicità tutti i comandi del 6502 saranno riportati nella seguente maniera:

AAA (#) $nn.

AAA, il comando vero e proprio, formato sempre da tre lettere come, ad esempio, LDA, STA, TAX ecc., è detto codice *mnemonico.*

Queste tre lettere sono seguite da un indirizzo o da un dato numerico, entrambi in notazione esadecimale. I dati si distinguono dagli indirizzi perchè hanno sempre il carattere "#" davanti al segno del dollaro ($).

Il comando più comune

Nei programmi scritti in BASIC la maggior parte delle istruzioni contengono comandi che fanno riferimento a variabili, sia per leggerne il valore sia per assegnargliene uno. Questo avviene anche in linguaggio macchina, con la differenza che non potremo leggere il contenuto di una variabile ma di una locazione. Questo significa che non avremo più a disposizione le comodità del BASIC. Infatti, in BASIC si assegna un nome alle variabili senza preoccuparsi di dove queste vengano effettivamente memorizzate. al contrario in linguaggio macchina, se vogliamo conservare dei dati dobbiamo scegliere una o più locazioni in cui memorizzarli, e prendere nota delle locazioni usate per fare in modo di non alterarli accidentalmente.

Fanno eccezione solamente i dati memorizzati nei registri interni (X,Y) del 6502 o nello STACK (ne abbiamo già parlato in precedenza). L'accumulatore "A" è un registro di passaggio nel quale memorizzeremo i dati sui quali effettuare le operazioni.
Il primo comando di cui parliamo è, pertanto, quello per caricare un dato nell'accumulatore.
In BASIC avremmo subito digitato:

A=10

per assegnare alla variabile A il valore 10.
In linguaggio macchina per caricare in A (accumulatore) il numero 10 si dà:

LDA # $0A

Questo comando vuol dire LoaD Accumulator (carica l'accumulatore). Notiamo il segno "#" per indicare che il numero seguente è un dato e non un indirizzo. Osserviamo anche che il 10 è stato scritto come 0A, quindi, in esadecimale. Come abbiamo già spiegato all'inizio di questa serie di articoli, è bene prendere l'abitudine di scrivere i numeri in notazione esadecimale poichè è molto più compatta di quella decimale.

Se, invece, avessimo voluto caricare nell'accumulatore il contenuto di un'altra locazione (ad esempio $A005) sarebbe stato sufficiente impostare il comando in questa maniera:

LDA $A005

I codici dei comandi

Naturalmente la notazione LDA $A005 non può essere compresa dal 6502, così com'è, a meno di utilizzare un Assemblatore di cui parleremo dopo.
Dovremo tradurre i comandi dati in Mnemonico (3 lettere) in codici formati da un solo byte.
Ad esempio il codice dell'istruzione LDA è $A9 se l'istruzione è seguita da un numero o $AD se è seguita da un indirizzo.

PERSONAL SOFTWARE

Impariamo il linguaggio macchina

Avremo cioè:

LDA # $0A = A9 0A
LDA $A005 = AD 05 A0

Come abbiamo già visto, il microprocessore capisce dal codice dell'istruzione quanti sono i byte seguenti che lo completano e come devono essere interpretati. Nel primo caso abbiamo una istruzione lunga 2 byte e nel secondo la lunghezza è di 3 byte.

Nel secondo esempio è importante notare che: *i numeri e gli indirizzi formati da due byte vengono memorizzati nell'ordine LSB-MSB.* Cioè, subito dopo il byte dell'istruzione viene il byte basso del numero e poi quello alto. (Ricordiamo che LSB - Least Significative Byte - sono gli otto bit a destra di un numero lungo 16 bit e omologamente MSB - Most Significative Byte - sono gli otto bit, cioè il byte, a sinistra).

Il primo programmino

È giunto ora il momento di scrivere le prime istruzioni in linguaggio macchina. Sommeremo al contenuto di una locazione di memoria il numero 20 (figura 1):

CLD	; Azzera il flag BCD
CLC	; Azzera il flag di Carry
LDA $0900	; Carica in A il contenuto della locazione $ 0900
ADC#$14	; Somma $14 (in decim. 20) al contenuto dell'accumulatore
STA $0900	; Rimette il risultato in %0900

In questi pochi comandi abbiamo utilizzato 4 nuove istruzioni:

CLD Serve ad azzerare il flag BCD. Se questo flag è settato (con SED) il 6502 rappresenta i dati in formato Binary Coded Decimal di cui abbiamo già parlato.

LC Azzera il flag di Carry che indica il riporto in una operazione di somma.

ADC Somma il numero contenuto nell'accumulatore al numero (o al contenuto della locazione) specificato dopo l'istruzione più il flag di carry. Da qui si vede l'importanza di azzerare tale flag prima di eseguire una addizione.

Se la somma provoca un riporto oltre l'ottavo bit il Carry viene settato altrimenti resta azzerato.

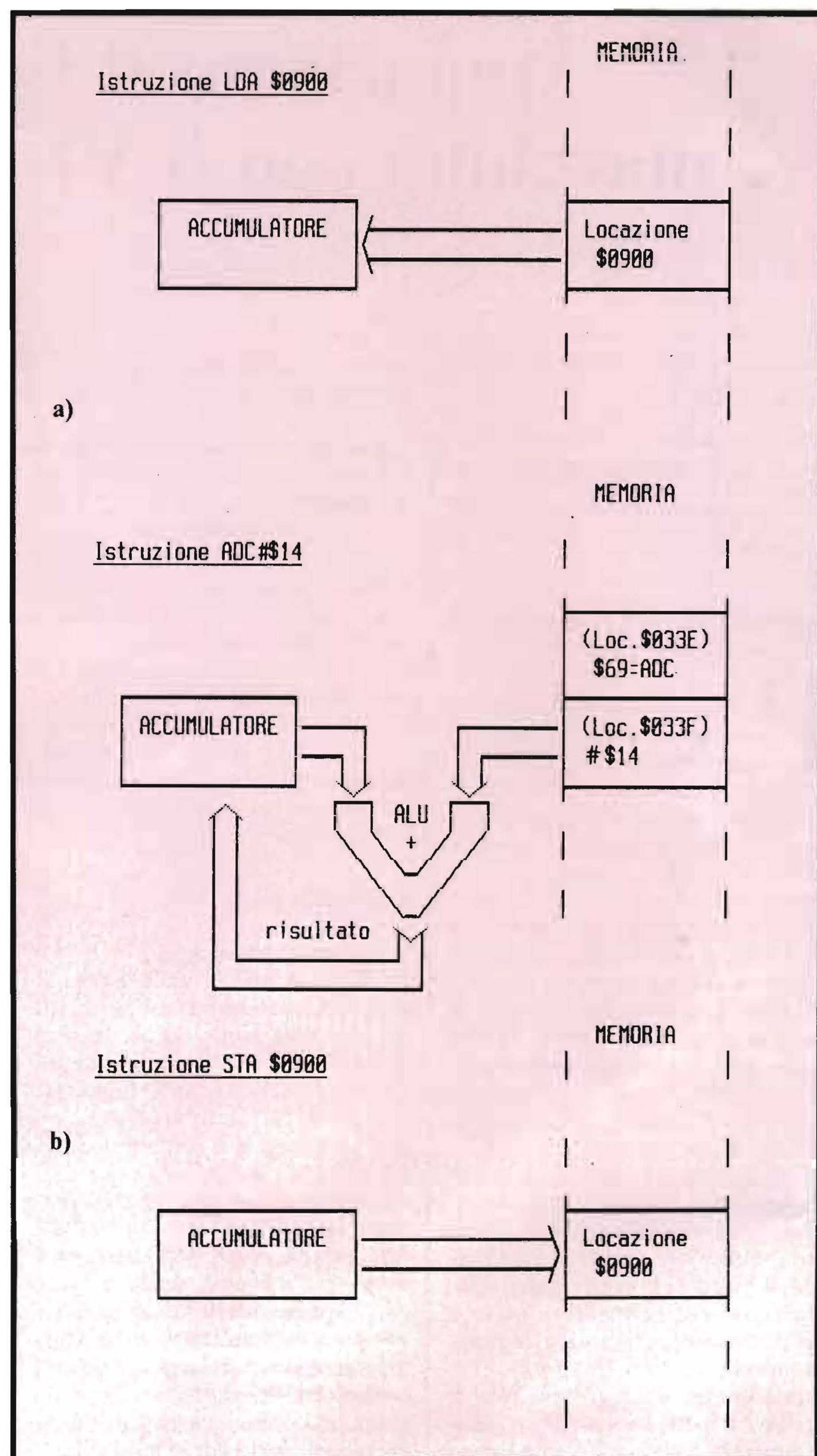


Figura 1. *Questa figura illustra il viaggio compiuto dai dati nel programma esempio (somma di due numeri da 1 byte) riportato nell'articolo.*

Impariamo il linguaggio macchina

STA È l'operazione esattamente opposta al LDA. Serve a memorizzare l'accumulatore nella locazione di memoria specificata dopo l'istruzione STA.
È importante notare, in questo breve esempio, la funzione svolta dall'accumulatore che si comporta come un registro di passaggio nel quale vengono effettuate tutte le operazioni, in questo caso una somma.

Per rendere utilizzabile questa routine scritta sarà necessario tradurla in codici indicando anche l'indirizzo di memoria in cui questi codici andranno inseriti.
Poichè, per ora ci serviremo del buffer di cassetta ($033C) avremo:

```
$033C  CLD         D8
$033D  CLC         18
$033E  LDA$0900    AD 00  09
$0341  ADC# $14    69  14
$0343  STA $0900   8D 00  09
```

Osserviamo che le istruzioni occupano sempre un byte di memoria più uno o due byte a seconda se sono seguite da un dato o dal numero di una locazione.

È possibile che l'istruzione sia seguita anche dall'indirizzo di una locazione in pagina zero. In questo caso è richiesto un solo byte oltre l'istruzione stessa. Ad esempio le due seguenti espressioni sono assolutamente identiche:

```
LDA  $00FE    AD  FE  00
LDA  $FE      A5  FE
```

Il secondo comando è, però, più conveniente in quanto occupa meno memoria ed è più veloce da eseguire.

Queste appena viste sono due delle possibili capacità di indirizzamento del 6502: indirizzamento assoluto (con un indirizzo a 16 bit) e indirizzamento in pagina zero (con un indirizzo ad otto bit riferito alla pagina zero della memoria). Presto vedremo gli altri modi.
Naturalmente con il tipo di indirizzamento cambia il codice dell'istruzione poichè è proprio questo ad indicare al 6502 come deve interpretare il dato seguente.

Memorizzazione dei programmi in linguaggio macchina

Resta ora il problema di mettere in memoria i codici relativi al programma in linguaggio macchina scritto.
Una soluzione è quella di tradurre gli indirizzi e i codici in decimale e poi immetterli in memoria con istruzioni POKE.
Ad esempio le prime due istruzioni del programma di somma di due numeri da un byte diventerebbero:

```
POKE828,216
POKE829,24
```

Naturalmente questo metodo va bene solo se i codici da introdurre in memoria sono pochi.

Una facilitazione è data dal programma del listato uno.
Questo permette di introdurre i codici direttamente in esadecimale e di controllare il contenuto della memoria.
Un ulteriore passo in avanti lo faremo con la presentazione di un assemblatore che permetterà di digitare i programmi in linguaggio macchina (detto anche Assembly) direttamente in Mnemonico.

Il programma in BASIC

Il programma presentato nel listato uno permette l'inserzione di codici in memoria direttamente in esadecimale.
Dato il RUN viene richiesta la locazione di inizio per la memorizzazione. A questo punto si può cominciare ad inserire i codici.
Questi devono essere formati da due caratteri esadecimali, e se si introduce più di un codice per linea, questi dovranno essere separati da uno spazio.
Ogni linea non può contenere più di 8 codici.
Il programma provvede anche a controllare i codici introdotti segnalando le linee contenenti dati errati.

Per effettuare la memorizzazione dei dati digitati è sufficiente premere il RETURN. Se tutti i codici sono corretti viene visualizzato il numero della successiva locazione di memoria e il programma resta in attesa di nuovi codici o di un comando.
I comandi a disposizione sono:
"★" termina il programma;
"$" torna a chiedere l'indirizzo di partenza. È utile se si vogliono caricare dei codici in zone non consecutive di memoria;
"!" visualizza i primi otto codici memorizzati dalla locazione di partenza in poi. Quindi se si preme un tasto qualsiasi vengono mostrati i contenuti delle otto locazioni successive altrimenti, premendo il tasto "HOME", si torna alla memorizzazione codici.

Perfezionamento del primo programma

Miglioriamo ora il programma di somma visto prima. Questa volta si vogliono sommare due numeri a 16 bit contenuti nelle locazioni $FA, $FB e $FC, $FD in pagina zero.
Naturalmente rispetteremo la convenzione dell'ordine byte basso-byte alto.
Il programma in Assembly sarà:

```
033C  CLD        D8       : Azzera Flag BCD
033D  CLC        18       : Azzera Flag Carry
033E  LDA  $FA A5   FA    : Carica LSB primo numero
0340  ADC  $FC 65   FC    : Somma LSB secondo numero
0342  STA  $FE 85   FE    : Mette il risultato in $FE
0344  LDA  $FB A5   FB    : Carica MSB primo numero
0346  ADC  $FD 65   FD    : Somma MSB secondo numero
0348  STA  $FF 85   FF    : Mette il risultato in $FF
0349  RTS        60       : RETURN
```

Come possiamo verificare sono cambiati i codici delle istruzioni LDA, ADC, STA. Questo perchè abbiamo usato l'indirizzamento in pagina zero. Da notare che all'indirizzo $033D viene pulito il flag di Carry, così che dopo la prima somma tale flag indicherà se vi è stato riporto. In tal caso il secondo ADC automaticamente aggiungerà il riporto alla somma dei byte alti.

L'ultima istruzione RTS è equiva-

Impariamo il linguaggio macchina

lente al comando BASIC "RETURN". Essa è necessaria sia al termine di una subroutine sia al termine di un programma intero in linguaggio macchina se si vuole tornare al BASIC.
Perchè il ritorno al BASIC sia effettuato senza problemi è necessario che il passaggio dal BASIC al linguaggio macchina sia stato eseguito con l'istruzione SYS o USR(X).

L'esecuzione di routine in linguaggio macchina da BASIC

Il metodo più semplice e più usato per passare dal BASIC al linguaggio macchina è senz'altro l'istruzione SYS (nnnn). Dove "nnnn" sta per l'indirizzo di partenza della routine linguaggio macchina in questione. Quando un programma BASIC incontra tale istruzione, interrompe l'esecuzione del programma principale e passa al programma in linguaggio macchina.

Il ritorno al BASIC si avrà quando verrà incontrata l'istruzione RTS (ReTurn from Subroutine).
Tornando al nostro programma di somma, dopo averlo caricato in memoria (nel buffer della cassetta), per vederlo in funzione scriveremo questo breve programma BASIC.

```
10 POKE 250,120:POKE 251,0:
   POKE 252,160:POKE 253,0
20 SYS 828
30 PRINT PEEK(254),PEEK(255)
```

Dando il RUN verranno sommati 120 e 160 e poichè il risultato è più grande di 256 troveremo 24 (byte basso) nella locazione $FE (254) e 1 (byte alto) in $FF (255).
La prossima volta vedremo molti altri comandi del 6502. ■

```
10 PRINT"▯CARICATORE DI CUDICI L.M.▮▮▮▮":OPEN1,0
20 PRINT:INPUT"INDIRIZZO PARTENZA:";A$:IFLEN(A$)<>4THEN10
25 PRINT"▮"
30 H$=LEFT$(A$,2):L$=RIGHT$(A$,2)
40 C$=H$:GOSUB500:AD%=R%*256:C$=L$:GOSUB500:AD%=AD%+R%:S%=AD%
60 HB%=AD%/256:LB%=AD%-HB%*256
70 R%=HB%:GOSUB1000:AD$=C$
80 R%=LB%:GOSUB1000:AD$=AD$+C$:IFRT=1THENRETURN
90 PRINT:PRINTAD$"   ";:INPUT#1,A$
100 IF LEN(A$)>30THENPRINT" ???";:GOTO90
105 IFA$="*"THENCLOSE1:END
107 IFLEFT$(A$,1)="!"THENU%=AD%:GOTO300
108 IFLEFT$(A$,1)="$"THEN20
110 E%=0:FORI=1TOLEN(A$):IFMID$(A$,I,1)=" "THEN116
112 K$=MID$(A$,I,1)
114 IFK$<"0"OR(K$>"9"ANDK$<"A")ORK$>"F"THENE%=1
116 NEXT
118 IFE%THENPRINT" ???";:GOTO90
160 FORCC=1TOLEN(A$)
170 C$=MID$(A$,CC,1):IFC$=" "THEN200
180 CC=CC+1:C$=C$+MID$(A$,CC,1)
190 GOSUB500:POKEAD%,R%:AD%=AD%+1
200 NEXT
210 GOTO60
300 RT=1:AD%=S%:PRINT"▮"
305 GOSUB60
310 PRINTAD$"  ";
320 FORI=0TO7
330 R%=PEEK(AD%+I):GOSUB1000:PRINTC$" ";
340 NEXT:PRINT
350 GETA$:IFA$=""THEN350
360 IFA$<>"▮"THENAD%=AD%+8:GOTO305
370 RT=0:AD%=U%:GOTO60
500 D$=LEFT$(C$,1):GOSUB700:R%=RS*16
510 D$=RIGHT$(C$,1):GOSUB700:R%=R%+RS
520 RETURN
700 RS=ASC(D$)-48+7*(ASC(D$)>58):RETURN
1000 H%=R%/16:L%=R%-H%*16
1010 C$=CHR$(H%+48-7*(H%>9))+CHR$(L%+48-7*(L%>9))
1020 RETURN
```

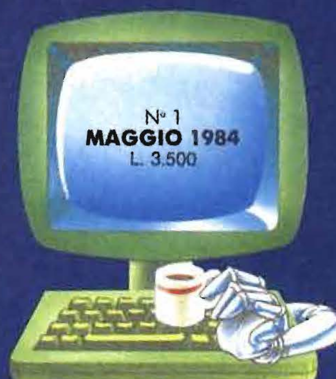
Listato 1. *Programma per immettere velocemente in memoria i codici delle istruzioni in linguaggio macchina.*

CORRI IN EDICOLA
C'È UNA NUOVA RIVISTA JACKSON
DEDICATA AL TUO
COMPUTER IN CASA.

UNA PUBBLICAZIONE DEL GRUPPO EDITORIALE JACKSON



hc



HOME COMPUTER

LA RIVISTA DEL COMPUTER IN CASA • PROGRAMMI • GIOCHI • NOVITÀ

Spedizione in Abb. Postale Gruppo III/70



I segreti dei Vincitori di Computer Play.

Tutto sul nuovo Sinclair.

16 Programmi in prova: Atari, Apple, CBM64, VIC 20, Spectrum, ZX 81, TI 99, Sharp MZ 700.

CONVIENE ABBONARSI ....
.... ALLE RIVISTE JACKSON

Uno sconto sicuro per chi sottoscrive un abbonamento

Riviste	Uscite	Importo globale	Costo abbonamento	Costo abbonamento estero
Personal Software	11	~~L. 44.000~~	L. 34.000	L. 51.000
Bit	11	~~L. 44.000~~	L. 35.000	L. 52.500
Informatica Oggi	11	~~L. 38.500~~	L. 31.000	L. 46.500
Elektor	12	~~L. 36.000~~	L. 29.000	L. 43.500
Automazione Oggi	11	~~L. 38.500~~	L. 30.500	L. 45.250
Elettronica Oggi	11	~~L. 44.000~~	L. 36.000	L. 54.000
L'Elettronica	22	~~L. 55.000~~	L. 44.000	L. 66.000
Telecomunicazioni Oggi	10	~~L. 35.000~~	L. 28.000	L. 42.000
Video Giochi	11	~~L. 38.500~~	L. 30.000	L. 45.000
Strumenti Musicali	10	~~L. 30.000~~	L. 24.000	L. 36.000
Home Computer	11	~~L. 38.500~~	L. 31.000	L. 46.000

Un supersconto a chi si abbona a due o più riviste

Tutti coloro che sottoscrivono l'abbonamento a due o più riviste godono di un prezzo ulteriormente agevolato, come appare nella seguente tabellina.

Esempio: Bit + Informatica Oggi
(Italia) L. 35.000 + L. 31.000 = L. 66.000 **meno** L. 2.000 = L. 64.000!
(Estero) L. 52.500 + L. 46.500 = 99.000 **meno** L. 2.000 = L. 97.000!

Abbonamento a 2 riviste: L. 2.000 **in meno** sulla somma dei 2 prezzi d'abbonamento
Abbonamento a 3 riviste: L. 4.000 **in meno** sulla somma dei 3 prezzi d'abbonamento
Abbonamento a 4 riviste: L. 7.000 **in meno** sulla somma dei 4 prezzi d'abbonamento
Abbonamento a 5 riviste: L. 10.000 **in meno** sulla somma dei 5 prezzi d'abbonamento
Abbonamento a 6 riviste: L. 13.000 **in meno** sulla somma dei 6 prezzi d'abbonamento
Abbonamento a 7 riviste: L. 16.000 **in meno** sulla somma dei 7 prezzi d'abbonamento
Abbonamento a 8 riviste: L. 20.000 **in meno** sulla somma degli 8 prezzi d'abbonamento
Abbonamento a 9 riviste: L. 25.000 **in meno** sulla somma dei 9 prezzi d'abbonamento
Abbonamento a 10 riviste: L. 30.000 **in meno** sulla somma dei 10 prezzi d'abbonamento
Abbonamento a 11 riviste: L. 35.000 **in meno** sulla somma degli 11 prezzi d'abbonamento

Per abbonarsi
Per sottoscrivere abbonamenti utilizzate il modulo di c.c.p. n. 11666203 intestato a Gruppo Editoriale Jackson - Via Rosellini, 12 - Milano, oppure inviate un assegno o un vaglia postale al nostro ufficio abbonamenti.

Othello per ZX81

— Parte seconda —

2° livello di gioco per lo ZX

di *Angelo Motta*

Sull'ultimo numero di **Personal Software** è stato presentato il programma Othello. Chi è già pratico del gioco si sarà sicuramente accorto, dopo alcune partite, che lo schema di gioco del computer è abbastanza semplice e, con un po' di pratica si riesce a ottenere facilmente la vittoria.

Come annunciato ecco un ampliamento del suddetto programma, che consente allo ZX di analizzare anche le possibili contromosse del giocatore e tenerne conto per determinare la propria mossa.

Modifiche apportate al programma originario

Le modifiche consistono sostanzialmente nell'aggiunta di una routine in linguaggio macchina che permette un 2° livello di gioco, e delle linee BASIC necessarie ad attivarla.

Cominceremo esaminando la routine illustrata nella figura 2, con la quale lo ZX sceglie la mossa al 2° livello. Inizialmente viene copiata la matrice di controllo del gioco in un'altra matrice creata nelle locazioni da 30820 a 30919 perchè, nello svolgimento della routine, la prima verrà modificata in continuazione. Viene creata una ulteriore matrice nelle locazioni da 31120 a 31219, contenente inizialmente tutti 0, che lo ZX utilizzerà per la scelta della mossa in base ai valori che inserirà in seguito alle valutazioni effettuate. A questo punto vengono passate in rassegna tutte le 64 caselle della scacchiera con lo stesso criterio seguito al livello 1: se la casella è occupata la scarta, se è libera chiama la routine di controllo della validità della mossa e, se questa è valida, diversamente dal 1° livello, provvede all'aggiornamento della matrice di controllo.

A questo punto lo ZX utilizza la routine con cui in precedenza sceglieva la mossa per controllare quale sia la miglior contromossa possibile in base al valore strategico delle caselle. La differenza tra il valore della

(16970)	33	100	120	1	100	0	17	0	120	237
	176	201	1	100	0	17	100	120	33	0
	120	237	176	33	143	121	6	100	35	54
	0	16	251	1	12	0	197	205	74	66
	193	197	33	255	119	9	126	254	0	194
(17020)	40	67	33	0	125	113	33	30	125	54
	0	33	10	125	54	1	33	20	125	54
	2	205	11	66	33	30	125	126	254	0
	202	40	67	193	197	0	33	129	64	9
	126	254	9	32	9	33	40	125	113	193
(17070)	205	74	66	201	89	1	8	0	197	33
	180	65	9	86	123	130	79	33	255	119
	9	126	254	1	32	29	121	130	79	33
	255	119	9	126	254	1	40	244	254	2
	32	13	121	146	79	33	255	119	9	54
(17120)	2	121	187	32	244	193	13	121	254	0
	32	202	33	10	125	54	2	33	20	125
	54	1	33	40	125	54	1	33	30	125
	54	1	205	209	65	33	40	125	126	254
	1	32	6	193	113	195	84	67	0	6
(17170)	0	79	33	129	64	9	86	193	197	33
	129	64	9	126	146	198	20	33	143	121
	9	119	193	12	121	254	90	194	110	66
	33	41	125	54	0	43	54	1	1	12
	0	33	143	121	9	126	254	0	40	10
(17220)	33	41	125	190	250	78	67	119	43	113
	12	121	254	90	32	231	33	10	125	54
	1	33	20	125	54	2	205	74	66	201

Figura 1. *Codici del linguaggio macchina da inserire nelle locazioni da 16970 a 17249 con il programma elencato nel listato 3.*

```
   2 REM (SEGUITA DA 280 CARATTE
RI)
  50 FOR I=16970 TO 17249
  55 INPUT A
  60 PRINT I;" = ";A
  65 POKE I,A
  70 IF PEEK 16418=2 THEN SCROLL

  75 NEXT I
  80 STOP
```

Listato 1. *Caricatore della routine in linguaggio macchina. Dare un RUN ed inserire i codici del listato 3.*

```
 245 PRINT ,,"LIVELLO DI GIOCO?
(1/2)"
 246 IF INKEY$<>"1" AND INKEY$<>
"2" THEN GOTO 246
 248 LET LV=CODE INKEY$
3870 LET L=USR (16829+(153 AND L
V=30))
```

Listato 2. *Modifiche da apportare al listato 1 del programma Othello presentato sul numero scorso.*

Othello per ZX81

casella che sta esaminando e quella della possibile contromossa avversaria, viene posto nella corrispondente casella della matrice creata nelle locazioni da 31120 a 31219. Dopo aver esaminato tutte le 64 caselle della scacchiera, lo ZX va a riprendere la suddetta matrice e sceglierà come mossa quella casella che conterrà il valore più alto, cioè la differenza fra mossa e contromossa più favorevole.

La strategia prevede inoltre che, se mentre lo ZX passa in rassegna le caselle trova come possibile mossa una casella d'angolo, obbligatoriamente la esegue in considerazione della importanza strategica della stessa. Inoltre, se nella fase di controllo della contromossa scopre che l'avversario non può muovere e deve passare il turno, interrompe la routine ed effettua quella mossa.

Con la possibilità di analizzare la mossa successiva dell'avversario, la qualità del gioco è notevolmente migliorata anche se non è possibile paragonarla a quella di programmi con più livelli di gioco e che permettono di analizzare più mosse successive.

Il listato ed i codici del linguaggio macchina da inserire nella REM 2 sono illustrati nella figura 1 e nel listato 1.

Si ricorda che tassativamente la REM 1 del programma originario dovrà essere seguita da 450 caratteri in modo che la routine per il 2° livello parta esattamente dalla locazione 16970; in caso contrario il programma non funzionerà e potrebbe addirittura bloccare la macchina.

La linea: 2870 LET L= URS (16829+(153 AND LV=30)) sostituisce la precedente e grazie agli operatori relazionali permette allo ZX di giocare al livello scelto, alle linee 245, 246.

Il programma al 2° livello è stato fatto provare a diversi giocatori e, mentre con i più esperti ha conseguito poche vittorie, ha rivelato di essere un avversario interessante per gli altri.

Figura 2. *Routine in linguaggio macchina mediante la quale lo ZX sceglie la mossa al 2° livello.*

Byte	Decimali			Hex			Mnemonici	Note
16970	33	100	120	21	64	78	LD HL, 30820	*Subroutine che riporta la*
16973	1	100	0	01	64	00	LD BC, 100	*matrice di controllo gioco*
16976	17	0	120	11	00	78	LD DE, 30720	*allo stato iniziale*
16979	237	176		ED	B0		LDIR	
16981	201			C9			RET	
16982	01	100	0	01	64	00	LD BC, 100	*salva le variabili di controllo*
16985	17	100	120	11	64	78	LD DE, 30820	*nelle locazioni da 30820*
16988	33	0	120	21	00	78	LD HL, 30720	*a 30919*
16991	237	176		ED	B0		LDIR	
16993	33	143	121	21	8F	79	LD HL, 31119	*azzera le locazioni che*
16996	06	100		06	64		LD B, 100	*conterranno i valori per*
16998	35			23			INC HL	*la scelta della mossa*
16999	54	0		36	00		LD (HL), 0	
17001	16	251		10	FB		DJNZ —5	
17003	1	12	0	01	0C	00	LD BC, 12	*inizio ciclo*
17006	197			C5			PUSH BC	
17007	205	74	66	CD	4A	42	CALL 16970	*ripristina matrice*
17010	193			C1			POP BC	*ripristina valore mossa*
17011	197			C5			PUSH BC	*nei registri BC*
17012	33	255	119	21	FE	77	LD HL, 30719	*la casella esaminata è*
17015	9			09			ADD HL, BC	*vuota?*
17016	126			7E			LD A, (HL)	*Se NO passa ad altra casella*
17017	254	0		FE	00		CP 0	
17019	194	40	67	C2	28	43	JP NZ 17192	
17022	33	0	125	21	00	7D	LD HL, 32000	*carica i byte con le variabili*
17025	113			71			LD (HL), C	*di controllo*
17026	33	30	125	21	1E	7D	LD HL, 32030	
17029	54	0		36	00		LD (HL), 0	
17031	33	10	125	21	0A	7D	LD HL, 32010	
17034	54	1		36	01		LD (HL), 1	
17036	33	20	125	21	14	7D	LD HL, 32020	
17039	54	2		36	02		LD (HL), 2	*e controlla la validità*
17041	205	11	66	CD	0B	42	CALL 16907	*della mossa*
17044	33	30	125	21	1E	7D	LD HL, 32030	*Se la mossa non è permessa*
17047	126			7E			LD A, (HL)	*passa ad altra casella*
17048	254	0		FE	00		CP 0	
17050	202	40	67	CA	28	43	JP Z 17192	
17053	193			C1			POP BC	*ripristino valore mossa*
17054	197			C5			PUSH BC	*nei registri BC*
17055	0			00			NOP	
17056	33	129	64	21	81	40	LD HL, 16513	*controlla se la casella*
17059	9			09			LD HL, BC	*è d'angolo*
17060	126			FE			LD A, (HL)	
17061	254	9		FE	09		CP 9	
17063	32	9		20	09		JR NZ +9	
17065	33	40	125	21	28	7D	LD HL, 32040	*se d'angolo effettua la*
17068	113			71			LD (HL), C	*mossa, ripristinando lo*
17069	193			C1			POP BC	*stato iniziale delle variabili*
17070	205	74	66	CD	4A	42	CALL 16970	*di controllo*
17073	201			C9			RET	
17074	89			59			LE E,C	*provvede all'aggiornamento*
17075	1	8	0	01	08	00	LD BC,8	*della scacchiera (routine*
17078	197			C5			PUSH BC	*simile a quella del*
17079	33	180	65	21	B4	41	LD HL, 16820	*listato 3).*
17082	9			09			ADD HL, BC	
17083	86			56			LD D, (HL)	
17084	123			7B			LD A, E	
17085	130			82			ADD A, D	
17086	79			4F			LD C, A	
17087	33	255	119	21	FE	77	LD HL, 30719	
17090	9			09			ADD HL, BC	
17091	126			7E			LD A, (HL)	
17092	254	1		FE	01		CP 1	
17094	32	29		20	1D		JR NZ + 29	
17096	121			79			LD A, C	
17097	130			82			ADD A, D	
17098	79			4F			LD C, A	
17099	33	255	119	21	FE	77	LD HL, 30719	
17102	9			09			ADD HL, BC	
17103	126			7E			LD A, (HL)	
17104	254	1		FE	01		CP 1	
17106	40	244		28	F4		JR Z-12	
17108	254	2		FE	02		CP 2	
17110	32	13		20	0D		JR NZ+13	
17112	121			79			LD A, C	
17113	146			92			SUB A, D	
17114	79			4F			LD A, C	
17115	33	255	119	21	FE	77	LD HL, 30719	
17118	9			09			ADD HL, BC	

Othello per ZX81

Seguito figura 2.

Byte	Decimali			Hex			Mnemonici	Note
17119	54	2		36	02		LD (HL), 2	
17121	121			79			LD A, C	
17122	187			BB			CP E	
17123	32	244		20	F4		JR NZ-12	
17125	193			C1			POP BC	
17126	13			0C			DEC C	
17127	121			79			LD A, C	
17128	254	0		FE	00		CP 0	
17130	32	202		20	CA		JR NZ-54	
17132	33	10	125	21	0A	7D	LD HL, 32010	*carica le variabili di controllo*
17135	54	2		36	02		LD (HL), 2	*con i valori per la mossa*
17137	33	20	125	21	14	7D	LD HL, 32020	*del giocatore*
17140	54	1		36	01		LD (HL), 1	
17142	33	40	125	21	28	7D	LD HL, 32040	
17145	54	1		36	01		LD (HL), 1	
17147	33	30	125	21	1E	7D	LD HL, 32030	
17150	54	1		36	01		LD (HL), 1	
17152	205	209	65	CD	D1	41	CALL 16849	*la routine di figura 3*
17155	33	40	125	21	28	7D	LD HL, 32040	*viene utilizzata per la*
17158	126			7E			LD A, (HL)	*contromossa del giocatore;*
17159	254	1		FE	01		CP 1	*se questi non può*
17161	32	6		20	06		JR NZ + 6	*muovere lo ZX sceglie*
17163	193			C1			POP BC	*la mossa che sta esaminando*
17164	113			71			LD (HL), C	
17165	195	84	67	C3	54	43	JP 17236	
17168	0			00			NOP	
17169	6	0		06	00		LD B, 0	*mette nella matrice nei*
17171	79			4F			LD C, A	*byte da 31120 a 31219*
17172	33	129	65	21	81	40	LD HL, 16513	*la differenza fra il va-*
17175	9			09			ADD HL, BC	*lore della mossa che sta*
17176	86			56			LD D, (HL)	*esaminando e la contro-*
17177	193			C1			POP BC	*mossa migliore per il*
17178	197			C5			PUSH BC	*giocatore. I valori sono*
17179	33	129	64	21	81	40	LD HL, 165113	*quelli indicati dalla*
17182	9			09			ADD HL, BC	*figura 1*
17183	126			7E			LD A, (HL)	
17184	146			92			SUB D	
17185	198	20		C6	14		ADD A, 20	
17187	33	143	121	21	8F	79	LD HL, 31119	
17190	9			09			ADD HL, BC	
17191	119			77			LD (HL), A	
17192	193			C1			POP BC	*carica la casella successiva*
17193	12			0C			INC C	*per l'esame della mossa*
17194	121			79			LD A, C	*e rieffettua il ciclo*
17195	254	90		FE	5A		CP 90	*da 17006*
17197	194	110	66	C2	6E	42	JP NZ	17006
17200	33	41	125	21	29	7D	LD HL, 32041	*byte di controllo per la*
17203	54	0		36	00		LD (HL), 0	*scelta della mossa*
17205	43			2B			DEC HL	
17206	54	1		36	01		LD (HL), 1	
17208	1	12	0	01	0C	00	LD BC, 12	*controlla che nelle loca-*
17211	33	143	121	21	8F	79	LD HL, 31119	*zioni da 31120 a 32219*
17214	9			09			ADD HL, BC	*il valore sia diverso da*
17215	126			7E			LD A, (HL)	*0 e, se si esamina il*
17216	254	0		FE	00		CP 0	*dato per la scelta della*
17218	40	10		28	0A		JR Z+10	*mossa*
17220	33	41	125	21	29	7D	LD HL, 32041	*se tale dato è minore di*
17223	190			BE			CP (HL)	*quello in carico passa al*
17224	250	78	67	FA	4E	43	JP M, 17228	*successivo*
17227	119			77			LD (HL), A	*carica il dato ed il numero*
17228	43			2B			DEC HL	*della mossa per il successivo*
17229	113			71			LD (HL), A	*confronto*
17230	12			0C			INC C	*carica la casella successiva per*
17231	121			79			LD A, C	*la scelta della mossa*
17232	254	90		FE	5A		CP 90	*e rieffettua il ciclo*
17234	32	231		20	E7		JR NZ-25	*da 17208 ripristina*
17236	33	10	125	21	0A	7D	LD HL, 32010	*le variabili di controllo*
17239	54	1		36	01		LD (HL), 1	*per lo ZX, la matrice*
17241	33	20	125	21	14	7D	LD HL, 32020	*per la gestione del*
17244	54	2		36	02		LD (HL), 2	*gioco allo stato iniziale*
17246	205	74	66	CD	4A	42	CALL 16970	*e ritorna*
17249	201			C9			RET	

Listato 3. *Codici della routine in linguaggio macchina per il secondo livello di gioco.*

```
16970 = 33
16971 = 100
16972 = 120
16973 = 1
16974 = 100
16975 = 0
16976 = 17
16977 = 0
16978 = 120
16979 = 237
16980 = 176
16981 = 201
16982 = 1
16983 = 100
16984 = 0
16985 = 17
16986 = 100
16987 = 120
16988 = 33
16989 = 0
16990 = 120
16991 = 237
16992 = 176
16993 = 33
16994 = 143
16995 = 121
16996 = 6
16997 = 100
16998 = 35
16999 = 54
17000 = 0
17001 = 16
17002 = 251
17003 = 1
17004 = 12
17005 = 0
17006 = 197
17007 = 205
17008 = 74
17009 = 66
17010 = 195
17011 = 197
17012 = 33
17013 = 255
17014 = 119
17015 = 9
17016 = 126
17017 = 254
17018 = 0
17019 = 194
17020 = 40
17021 = 67
17022 = 33
17023 = 0
17024 = 125
17025 = 113
17026 = 33
17027 = 30
17028 = 125
17029 = 54
17030 = 0
17031 = 33
17032 = 10
17033 = 125
17034 = 54
17035 = 1
17036 = 33
17037 = 20
17038 = 125
```



Othello per ZX81

Seguito listato 3.

```
17039 = 54
17040 = 2
17041 = 205
17042 = 11
17043 = 66
17044 = 33
17045 = 30
17046 = 125
17047 = 126
17048 = 254
17049 = 0
17050 = 202
17051 = 40
17052 = 67
17053 = 193
17054 = 197
17055 = 0
17056 = 33
17057 = 129
17058 = 64
17059 = 9
17060 = 126
17061 = 254
17062 = 9
17063 = 32
17064 = 9
17065 = 33
17066 = 40
17067 = 125
17068 = 113
17069 = 193
17070 = 205
17071 = 74
17072 = 66
17073 = 201
17074 = 89
17075 = 1
17076 = 6
17077 = 0
17078 = 197
17079 = 33
17080 = 180
17081 = 66
17082 = 9
17083 = 86
17084 = 123
17085 = 130
17086 = 79
17087 = 33
17088 = 255
17089 = 119
17090 = 9
17091 = 126
17092 = 254
17093 = 1
17094 = 32
17095 = 29
17096 = 121
17097 = 130
17098 = 79
17099 = 33
17100 = 255
17101 = 119
17102 = 9
17103 = 126
17104 = 254
17105 = 1
17106 = 40
17107 = 244
17108 = 254
17109 = 2
```

Seguito listato 3.

```
17110 = 32
17111 = 13
17112 = 121
17113 = 146
17114 = 79
17115 = 33
17116 = 255
17117 = 119
17118 = 9
17119 = 54
17120 = 2
17121 = 121
17122 = 187
17123 = 32
17124 = 244
17125 = 193
17126 = 13
17127 = 121
17128 = 254
17129 = 0
17130 = 32
17131 = 202
17132 = 33
17133 = 10
17134 = 126
17135 = 54
17136 = 2
17137 = 33
17138 = 20
17139 = 125
17140 = 54
17141 = 1
17142 = 33
17143 = 40
17144 = 125
17145 = 54
17146 = 1
17147 = 33
17148 = 30
17149 = 125
17150 = 54
17151 = 1
17152 = 205
17153 = 209
17154 = 65
17155 = 33
17155 = 40
17157 = 125
17158 = 126
17159 = 254
17160 = 1
17161 = 32
17162 = 6
17163 = 193
17164 = 113
17165 = 195
17166 = 84
17167 = 67
17168 = 0
17169 = 6
17170 = 0
17171 = 79
17172 = 33
17173 = 129
17174 = 64
17175 = 9
17176 = 86
17177 = 193
17178 = 197
17179 = 33
```

Seguito listato 3.

```
17180 = 129
17181 = 64
17182 = 9
17183 = 126
17184 = 146
17185 = 198
17186 = 20
17187 = 33
17188 = 143
17189 = 121
17190 = 9
17191 = 119
17192 = 193
17193 = 12
17194 = 121
17195 = 254
17196 = 90
17197 = 194
17198 = 110
17199 = 55
17200 = 33
17201 = 41
17202 = 125
17203 = 54
17204 = 0
17205 = 43
17206 = 54
17207 = 1
17208 = 1
17209 = 12
17210 = 0
17211 = 33
17212 = 143
17213 = 121
17214 = 9
17215 = 126
17216 = 254
17217 = 0
17218 = 40
17219 = 10
17220 = 33
17221 = 41
17222 = 125
17223 = 190
17224 = 250
17225 = 78
17226 = 67
17227 = 119
17228 = 43
17229 = 113
17230 = 12
17231 = 121
17232 = 254
17233 = 90
17234 = 32
17235 = 231
17236 = 33
17237 = 10
17238 = 125
17239 = 54
17240 = 1
17241 = 33
17242 = 20
17243 = 125
17244 = 54
17245 = 2
17246 = 205
17247 = 74
17248 = 66
17249 = 201
```

Robot a caccia

Un inseguimento mozzafiato con lo Spectrum

di *Marcello Morchio*

Il programma che vi propongo è un gioco, la cui meccanica è molto semplice, ma che contiene vari algoritmi interessanti e una routine in linguaggio macchina tratta proprio da **Personal Software**.
Il funzionamento del gioco è molto semplice: lo schermo è sempre lo stesso e consiste in un campo minato circondato dal fuoco.
Il giocatore comanda un omino che appare inizialmente al centro del campo mentre nell'angolo in alto a sinistra appare un robot rosso che lo insegue. Lo scopo del gioco è di fare in modo che il robot calpesti il maggior numero di mine possibile e, naturalmente senza farsi raggiungere, il che comporterebbe la perdita di una delle tre vite. Il punteggio viene visualizzato solo alla fine del gioco, e se è superiore a 1000 (molto difficile) si ha una particolare onorificenza che vedremo in seguito.

Le routine principali

Le routine principali del programma sono 6.
La prima è come al solito l'inizializzazione delle variabili, a cui segue l'inizializzazione dello schermo che in questo caso è particolarmente importante, perché, oltre a stabilire i valori per inchiostro, carta e bordo (rispettivamente 0,4 e 2), presenta il ciclo di posizionamento delle mine e la memorizzazione delle loro posizioni.
Viene poi il loop principale, dove viene letta la tastiera, vengono aggiornate le coordinate dell'omino del giocatore e, basandosi su queste ultime, viene mosso il robot. Sempre in questo loop vengono stampati il robot e l'omino e viene controllato che quest'ultimo non esca dallo schermo o investa una mina.
Le ultime tre routine riguardano la fine del gioco o il lampeggiamento dello schermo, e sono le più semplici, ma sono quelle che chiamano la routine in linguaggio macchina già accennata, della quale parlerò nel prossimo paragrafo.

La routine in linguaggio macchina

Questa routine è tratta da **Personal Software** n. 5 da *"I segreti dei personal"* dello Spectrum, a pag. 48, ma ha subito alcune modifiche.
Per coloro che non possedessero il suddetto numero della rivista vediamo comunque l'uso e la codifica in Assembly della routine.
Essa serve per ottenere il mutamento degli attributi dello schermo senza dover usare CLS, INK, PAPER e BORDER.
La routine lavora nella parte del display file riservata agli attributi (indirizzi da 22528 a 23296), scrivendovi dentro i valori stabiliti dall'utente, secondo lo schema della funzione ATTR (128 per il lampeggio (0 se no lampeggio) + 64 per la luminosità + 8 moltiplicato per il valore della carta + il valore dell'inchiostro).

```
ld hl, 22528        33 0 88
ld bc, 704          1 192 2
ld (hl), 7          54 7
dec bc              11
ld a, b             120
or c                177
ret z               200
inc hl              35
jr, − 8             14 247
```

Il "7" della terza istruzione (ld (hl), 7) è il valore che può essere cambiato per ottenere i vari effetti di colore (e lampeggio) secondo lo schema già descritto. Esso può essere cambiato facendo una POKE all'indirizzo della prima istruzione + 7.
Coloro che hanno già visto questo programma su **Personal Software**, noteranno che nella seconda istruzione, c'è un 704 invece del 768 dell'originale. Questo perché col 768 veniva colorata anche la parte bassa dello schermo (le due linee a disposizione del sistema), mentre col 704 si colorano solo le 22 linee usabili direttamente con PRINT. La ragione di questo cambiamento è che nel nostro caso dando un colore uniforme in tutto lo schermo il giocatore avrebbe creduto che anche le due ultime linee fossero parte del campo da gioco, mentre il computer avrebbe interpretato il fatto come un tentativo di sconfinare dalle 22 linee stampabili con gravi conseguenze per il giocatore.

Strategia di gioco

Un paragrafo come questo sarebbe più adatto per una rivista come *Video Giochi* più che per **Personal Software** ma già che il gioco c'è tanto vale giocarlo bene.
Come già detto lo scopo del gioco e far calpestare le mine dal robot nemico; ciò non è solo fatto di fortuna, perché, come è logico, il robot continua a inseguirci. Quando passa su una mina la cancella, e lo schermo ha un "sobbalzo", ottenuto invertendolo e poi riportandolo alla normalità velocemente, natural-

mente con l'ausilio della routine in linguaggio macchina. Una buona strategia può essere quella di, all'inizio della partita, restar fermi e attendere che il robot tolga le mine più vicine a noi, perché raramente parte subito in rotta di collisione, e rimane a "orbitare" con un andamento pseudo casuale intorno all'omino. Seguendo questa tecnica bisogna aspettare che il robot ci sia piuttosto lontano prima di partire, perché fa presto a correggere la rotta venendoci addosso. Durante il gioco capita che mentre si percorre un tratto in linea retta, il robot ci venga talmente vicino da sfiorarci ad ogni suo attacco, e producendo ogni volta un effetto come se avesse colpito una mina. Questo è il momento migliore per far punti, perché, oltre a pulire le mine vicino a noi, il robot ci fa guadagnar punti ogni volta che ci passa vicino. Per sfruttare a pieno questa situazione bisogna, alla fine del rettilineo, invertire repentinamente la direzione quando ci è ancora distan-

```
   1 CLEAR 32499: LET r=32499
   2 FOR i=r+1 TO r+15
   5 READ m
   6 POKE i,m
   7 NEXT i
   8 DATA 33,0,88,1,192,2,54,7,1
1,120,177,200,35,24,247
   9 GO SUB 8000
  10 LET x=10: LET y=15: LET a=0
LET b=0
  20 LET sx=0: LET sy=0: LET sa=
0: LET sb=1
  30 LET p=0: DIM n(60): DIM m(6
0)
  40 LET cambio=32500
  50 LET p$="C": LET v$="888"
  60 LET v=3
 100 BORDER 2: INK 1: PAPER 4: C
LS
 110 FOR n=1 TO 60: LET n(n)=INT
 (RND*21)+1: LET m(n)=INT (RND*3
1)+1: PRINT AT n(n),m(n);"D": NE
XT n
 200 IF INKEY$="" THEN GO TO 300
 210 LET a$=INKEY$
 220 IF a$="5" THEN LET sy=-1: L
ET sx=0: LET p$="A"
 230 IF a$="6" THEN LET sx=1: LE
T sy=0: LET p$="C"
 240 IF a$="7" THEN LET sx=-1: L
ET sy=0: LET p$="C"
 250 IF a$="8" THEN LET sy=1: LE
T sx=0: LET p$="B"
 300 LET x=x+sx: LET y=y+sy
 310 IF x>21 OR x<0 OR y>31 OR y
<0 THEN LET x=x-sx*2: LET y=y-sy
*2
 315 IF sx=0 AND sy=0 THEN GO TO
 330
 320 IF SCREEN$ (x,y)<>" " THEN
GO SUB 1000
 330 IF a=x AND y=b THEN GO SUB
1000
 335 LET h=INT (RND*2)
 340 IF h=1 THEN GO TO 360
 350 IF x>a THEN LET sa=1
 355 IF x<a THEN LET sa=-1
 360 IF h=0 THEN GO TO 380
 370 IF y>b THEN LET sb=1
 375 IF y<b THEN LET sb=-1
 380 LET a=a+sa: LET b=b+sb
 390 LET a=a-(a>21)+(a<0): LET b
=b+(b<0)-(b>31)
 410 IF SCREEN$ (a,b)<>" " THEN
GO SUB 800
 420 PRINT AT a-sa,b-sb;" ";AT a
,b; INK 2;"F"
 425 IF sx=0 AND sy=0 THEN GO TO
 440
 430 IF SCREEN$ (x,y)<>" " THEN
GO SUB 1000
 440 IF a=x AND y=b THEN GO SUB
1000
 445 PRINT AT x-sx,y-sy;" ";AT x
,y; INK 0;p$
 460 BEEP .01,-10: BEEP .01,10
 465 PRINT AT 0,0; INK 7;v$( TO
v)
 470 GO TO 200
 500 CLS
 510 FOR n=1 TO 117: PRINT "BRAV
O!";: BEEP .007,13: NEXT n
 520 FOR n=0 TO 255: POKE cambio
+7,n
 525 LET i=USR cambio
 530 BEEP .009,(RND*10)-5
 540 PRINT AT 10,7; FLASH 1; PAP
ER 7;"PUNTEGGIO=";p;AT 12,6;"SEI
 GRANDE!!"
 550 IF INKEY$<>"" THEN RUN 10
 560 NEXT n: GO TO 520
 600 PRINT AT x,y;" ": PRINT AT
x-sx,y-sy;" "
 605 PRINT AT 0,0;"   "
 610 RETURN
 800 POKE cambio+7,7: RANDOMIZE
USR cambio
 810 LET p=p+21
 820 POKE cambio+7,33: RANDOMIZE
 USR cambio: RETURN
1000 LET v=v-1: IF v<>0 THEN LET
 a=0: LET b=0: POKE cambio+7,7:
RANDOMIZE USR cambio: PRINT AT x
-sx,y-sy; INK 7; PAPER 0;"E": BE
EP .5,10: POKE cambio+7,33: RAND
OMIZE USR cambio: GO SUB 600: RE
TURN
1005 FOR n=1 TO 3: FOR m=0 TO 7:
 POKE cambio+7,m
1010 RANDOMIZE USR cambio: BEEP
.01,10: BEEP .01,50
1020 NEXT m: NEXT n
1025 IF p>=1000 THEN GO TO 500
1030 PRINT AT 10,7;"PUNTEGGIO=";
p
1040 FOR n=1 TO 7: POKE cambio+7
,n
1050 RANDOMIZE USR cambio
1060 BEEP .05,n
1070 IF INKEY$<>"" THEN RUN 10
1080 NEXT n: GO TO 1040
8000 FOR n=USR "a" TO USR "f"+7
8010 READ m: POKE n,m
8020 NEXT n: RETURN
8030 DATA 24,24,20,124,8,20,34,1
38,24,24,80,62,16,40,68,195,48,4
8,124,176,56,40,40,32,60,126,223
,191,191,223,102,60,17,138,80,41
,144,44,67,135,60,90,60,231,24,3
6,36,102
9000 CLEAR : SAVE "ROBOT" LINE 1
```

Listato 1. *Programma Robot. I caratteri grafici usati sono: "A" = omino a sinistra; "B" = omino a destra; "C" = omino in alto, "D" = mina; "E" = esplosione; "F" = robot.*
Il listato contiene i caratteri grafici originali (prima dell'inizializzazione) per evitare confusione in fase di battitura. Ricordarsi di usare il modo grafico (cursore "G") per inserire le lettere "A", "B", "C", "D", "E", "F" (maiuscole) in tutte le linee dove compaiono tra apici, isolate o a gruppi di 3. In particolare si faccia attenzione alle linee 50, 110, 220, 230, 240, 250, 420, 1000. Alla linea 8000 si usino le lettere minuscole normali.

te, per poi continuare nella stessa rotta della andata. Bisogna però non far andare troppo per le lunghe questa strategia, perché prima o poi il robot ci distrugge.

Altro metodo per far punti è quello di appostarsi dietro a un grumo di mine, in modo che il robot gli passi sopra per raggiungerci.

Se alla fine delle tre vite, il cui numero è indicato in alto a sinistra con degli omini bianchi in fila, il punteggio è superiore a 1000, lo schermo viene coperto di scritte "BRAVO!" e mentre appare il punteggio lampeggiante, le scritte e il fondo cambiano colore, luminosità e stato di lampeggio, in un sottofondo di scalette musicali. Se invece non si sono raggiunti i 1000 punti, il fondo dello schermo diventa nero e il colore delle scritte oscilla continuamente dal nero al giallo. Sempre con un sottofondo di scalette musicali, in ambedue i casi il computer aspetta che venga premuto un tasto per ricominciare da capo il gioco.

Un altro piccolo consiglio: se la perdita di una vita è causata dal robot non c'è niente da fare ma se la causa del decesso è una mina, e se dietro a questa mina ce ne sono altre, bisogna cambiare velocemente direzione, perché, anche se il robot riparte dall'angolo a destra in alto, l'omino continua dal posto in cui è morto, continuando nella stessa direzione. Un'ultima nota: per eludere il robot bisogna ingannarlo con finte veloci, procedendo in modo da farsi superare per avere il tempo di cambiare direzione mentre è distante.

Una curiosità: il punteggio viene realizzato anche se il robot urta uno degli omini che indicano le vite o anche uno dei robot che lascia ai bordi oppure quando finisce una manche.

Un'altra particolarità.

Il nostro omino è sui pattini. Cioè appena viene premuto uno dei tasti di movimento (5, 6, 7, 8) l'omino parte e non si ferma più per tutto il resto del gioco.

Note al listato

All'inizio (linee da 1 a 8) viene caricata la routine in linguaggio macchina (notate che queste linee sono le stesse del listato 2 pubblicato sul n. 5 di **Personal Software** e possono essere usate anche da sole per caricare la routine). Segue un GO SUB 8000 che chiama la routine che inizializza i caratteri grafici. Dalla linea 10 alla linea 60 troviamo l'inizializzazione delle variabili principali. Da notare la variabile "cambio" che contiene l'indirizzo d'inizio della routine linguaggio macchina e la stringa v$ contenente i tre omini che indicano il numero delle vite rimaste.

La linea 100 e la linea 110 inizializzano lo schermo e stampano le mine memorizzando la loro posizione per un motivo che vedremo dopo.

Alla linea 200 inizia il gioco con un'istruzione che fa saltare la parte che legge la tastiera se non è premuto alcun tasto.

È questo che permette di mantenere la stessa direzione se non vengono premuti tasti. Si noti che la variabile stringa p$ contiene l'omino nelle tre posizioni che può assumere a seconda del tasto premuto.

La linea 310 fa scattare indietro l'omino se raggiunge i limiti dello schermo ma non ne cambia la direzione (il che si può comunque attuare aggiungendo alla 310:

LET sx = − sx : LET sy = − sy).

Dalla linea 335 inizia la routine che fa muovere il robot impazzito. Questa routine è una di quelle apparse su **Personal Software** n. 10-11, nell'articolo *Giochi di inseguimento*.

Le linee 420 e 445 stampano il robot e l'omino nelle loro posizioni (a, b per il robot; x, y per l'omino) e degli spazi bianchi nelle vecchie coordinate.

Infine la linea 470 chiude il ciclo con un GOTO 200.

Dalla linea 500 inizia la routine che riempie lo schermo di "BRAVO" che si mette in funzione se il punteggio è superiore a 1000. La routine 600 è una routine "per sbaglio", creata per non caricare troppo la linea 1000 che esamineremo poi.

Le linee da 800 a 820 fanno fare un lampeggio allo schermo se il robot calpesta una mina.

La routine alla linea 1000 è caricata tutta su una linea perché è stata fatta alla fine della stesura del programma, dopo la decisione di dare all'omino tre vite invece di una, secondo l'idea originaria. Così la linea 1000 fa lampeggiare lo schermo quando finisce una vita, stampa il carattere dell'esplosione, simula l'esplosione e riporta al programma principale se non sono finite le vite.

Viene quindi controllato se il punteggio è maggiore di 1000, il che comporterebbe un salto alla linea 500, e dopo viene stampato il punteggio e si attende che venga premuto un tasto.

Dopo la routine che inizializza i caratteri grafici si trova la linea 9000 che salva il programma.

Eventuali ampliamenti del programma

Ecco finalmente spiegata la ragione della memorizzazione delle posizioni delle mine: aggiungendo la linea:

1025 FOR n = 1 TO 60 : PRINT AT n (n) m (n); "E" (E grafica, cioè il carattere dell'esplosione) : BEEP .007, n − 10 : NEXT n

si ottiene l'esplosione di tutte le mine alla morte del terzo omino. Francamente non mi sembra molto bello, ma con colori opportuni (io uso un TV in bianco e nero), si possono ottenere effetti interessanti.

Un'altra modifica potrebbe essere di aumentare il numero di mine, cambiando tutti i 60 delle linee 30 e 110 con il numero scelto. Sarebbe anche interessante fare una specie di labirinto con le mine, anche se occorrerebbe modificare molto la parte dell'inizializzazione dello schermo, ma si otterrebbe un gioco molto nuovo. ■

OLTRE L'ORIZZONTE CON LO SPECTRUM





77 PROGRAMMI PER SPECTRUM
150 Pagine. 30 illustrazioni a colori
Cod. 555 A
L. 16000



GRUPPO EDITORIALE JACKSON

E PER LO ZX81...

66 PROGRAMMI PER ZX81 E ZX80 CON NUOVA ROM + HARDWARE

144 Pagine
Cod. **520 D**
L. **12000**





Grafica avanzata sul TI 99/4A

Utilizziamo gli sprite mediante l'Extended BASIC

di *Filippo Cerulo*

La caratteristica forse più interessante dell'Extended BASIC implementato sul TI 99/4A è la possibilità di controllare da programma, in modo relativamente semplice, sagome in movimento sullo schermo. Il nome di queste sagome è sprite (che tradotto in modo letterale significa folletto), nome che ne denuncia l'origine essenzialmente ludica (nel senso di videogame).
In verità poi ci si accorge subito che uno sprite può essere molto utile anche in applicazioni che col gioco hanno poco a che vedere, ma ciò non toglie che la prima immagine che ci viene in mente è quella dell'astronave che spara agli invasori.

È meglio dire subito che (a parte il nome) i nostri sprite hanno poco in comune con quelli del Commodore 64 e sono fondamentalmente diversi dal Player-Missile implementato sui computer Atari. Ciò soprattutto per una precisa scelta tecnica: niente bit da settare con relative PEEK e POKE, niente mappe di memoria da consultare, niente locazioni misteriose da scoprire, ma solo sottoprogrammi su ROM pronti all'uso e direttamente richiamabili da BASIC, con eventuale passaggio di parametri. Naturalmente, non tutti possono essere d'accordo con questa filosofia, ma mi sembra la scelta più logica, se consideriamo la struttura del computer e soprattutto dell'ottimo interprete BASIC. Il rovescio della medaglia è una certa lentezza di alcune istruzioni, ma niente di irreparabile.
D'altronde non si può avere tutto dalla vita e dal ... personal.

```
200 CALL CLEAR :: CALL SCREEN (15) :: DIM MU (24)
210 CALL MAGNIFY (2) :: CALL CHAR (132, "60F0F8FCF8F06",
    133, "000000F", 131, RPT$ ("F", 16), 129, "0", 130, RPT$ ("0F", 8))
    :: CALL COLOR (13, 7, 1)
220 FOR I = 2 TO 19 :: MU (I) = 2 :: NEXT I
230 K = 1 :: VE = 12 :: KA = 0 :: COL = 20
240 J = 0 :: CALL VCHAR (1, COL-1, 32, 24)
250 FOR I = 2 TO 19 :: IF MU (I) > 0 THEN CALL HCHAR (I, COL,
    129 + MU (I)) :: J = J + 1
260 NEXT I :: IF J = 0 THEN 370
270 CALL SPRITE (# 2, 132, 5, 233, 17) :: IF K > 9 THEN K = K - 10
    :: KA = KA + 1 :: CALL SPRITE (# 4, KA + 48, 5, 169, 65)
280 CALL SPRITE (# 5, K + 48, 5, 169, 81) :: CALL MOTION (# 2,
    VE, 0)
290 CALL KEY (5, COD, ST) :: IF COD < > 32 THEN 350
300 CALL POSITION (# 2, RI, CO) :: IF RI > 157 THEN 350
310 CALL SPRITE (# 3, 133, 5, RI, CO, 0, 90) :: P = INT ((RI + 6)/8)
    + 1
320 FOR J = 1 TO 80 + (COL - 20) ★ 10 :: NEXT J
330 IF MU (P) > 0 THEN MU (P) = MU (P) - 1 :: CALL HCHAR (P,
    COL 129 + MU (P)) :: CALL SOUND (100, - 5, 3)
340 CALL DEL SPRITE (# 3)
350 CALL COINC (# 2, 190, 17, 40, S) :: IF S = 0 THEN 290
360 CALL DEL SPRITE (# 2) :: COL = COL - 1 ★ (COL < 28) :: K =
    K + 1 :: VE = VE - 1 ★ (VE < 18) :: GOTO 240
370 CALL DEL SPRITE (ALL) :: DISPLAY AT (3, 2) ERASE ALL :
    "MURO ABBATTUTO IN"
380 CALL SPRITE (# 2, KA + 48, 5, 41, 33, # 3, K + 47, 5, 41, 49)
390 DISPLAY AT (10, 2) : "TENTATIVI." :: DISPLAY AT (15, 2)
    BEEP : "VUOI GIOCARE ANCORA ? (S/N)"
400 CALL KEY (5, COD, ST) :: IF ST = 0 THEN 400
410 IF COD = 83 THEN CALL CLEAR :: CALL DEL SPRITE (ALL) ::
    GOTO 220
420 CALL CLEAR :: STOP
```

Figura 1. *Il listato del programma Darts.*

Folletti

Il nostro sprite è una entità indipendente dal contesto in cui opera. Fondamentalmente possiede i seguenti attributi: una sagoma, un colore, una posizione, una velocità (se è in movimento). È possibile modificare uno qualsiasi di questi attributi, indipendentemente dagli altri ed in qualunque momento, tramite istruzioni dedicate. Possiamo inoltre rilevare da programma alcune informazioni che lo riguardano: la posizione, la sua distanza da un punto dello schermo o da un altro sprite, la sovrapposizione a punti dello schermo o ad altri sprite. Ogni sprite è

Grafica avanzata sul TI 99/4A

identificato da un numero e nelle istruzioni che lo riguardano si fa riferimento all'angolo in alto a sinistra della sagoma che lo rappresenta.
Per uno sprite lo schermo è formato da 256 righe e 256 colonne (indicate nel manuale col prefisso "dot"), ma solo le prime 193 righe sono visualizzabili.

L'istruzione per far nascere uno sprite è:
- CALL SPRITE (# numero di sprite, codice di carattere, colore, riga, colonna, [velocità di riga, velocità di colonna]);
- il "numero di sprite" è un intero tra 1 e 28 compresi, usato come identificatore. In caso di coincidenza sullo schermo, lo sprite di numero inferiore si sovrappone all'altro;
- il "codice di carattere" è il codice ASCII del carattere di cui vogliamo che lo sprite assuma la sagoma, cioè un intero tra 32 e 143;
- il colore è il codice del colore che lo sprite dovrà assumere (un intero tra 1 e 16);
- le "riga, colonna" sono le coordinate della posizione di partenza dello sprite (due interi tra 1 e 256);
- le "velocità di riga, velocità di colonna" sono opzionali e rappresentano l'eventuale movimento da imprimere allo sprite.

(Interi tra − 128 e + 127).

Il seguente mini-programma crea uno sprite con la sagoma del carattere "A" (cod. ASCII 65), di colore blu, che si muove in verticale (fermare l'esecuzione con CLEAR):

```
200 CALL CLEAR
210 CALL SPRITE (# 2, 65, 5, 2, 42, 10, 0)
220 GOTO 220
```

Adesso esamineremo, un po' più da vicino ed uno per uno, gli attributi di uno sprite.

La forma del folletto

Fondamentalmente, per scegliere una sagoma per il folletto si opera ridefinendo caratteri in modo usuale. Uno sprite può avere quattro "grandezze", in corrispondenza a quattro parametri usati nell'istruzione: CALL MAGNIFY (parametro).
In particolare può essere:
a) di un solo carattere (valore di default): CALL MAGNIFY (1);
b) di un solo carattere ingrandito: CALL MAGNIFY (2);
c) di quattro caratteri: CALL MAGNIFY (3);
d) di quattro caratteri ingrandito: CALL MAGNIFY (4).

Nei casi a) e b) la sagoma dello sprite è quella del carattere il cui codice compare nella CALL SPRITE, ingrandita o meno.
Nel seguente programma ridefiniamo il carattere "A" come un ometto che dopo un po' diventa più grande grazie alla istruzione della riga 240 (fermare il programma con CLEAR):

```
200 CALL CLEAR
210 CALL CHAR (65, "1818FF3C3C242424")
220 CALL SPRITE (# 2, 65, 5, 2, 42, 10, 0)
230 FOR I = 2 TO 1000 :: NEXT I
240 CALL MAGNIFY (2)
250 GOTO 250
```

Con una CALL MAGNIFY (2) la sagoma dello sprite rimane invariata, ma le sue dimensioni raddoppiano, cioè lo spazio occupato è quello corrispondente a quattro caratteri.

Nei casi c) e d), invece, la sagoma dello sprite è formata dall'unione di quelle di quattro caratteri. In particolare dal carattere specificato nella CALL SPRITE e dai tre caratteri seguenti, se il codice del primo carattere (quello specificato) è divisibile per quattro. In caso contrario la "conta" comincia col primo carattere, precedente il codice specificato, il cui codice sia divisibile per quattro.
La corrispondenza delle sagome procede dall'alto in basso e da destra a sinistra.
Ecco un esempio (fermare il programma con CLEAR):

```
200 CALL CLEAR
210 CALL MAGNIFY (3)
220 CALL SPRITE (# 2, 68, 5, 2, 42, 9, 0)
230 FOR I = 2 TO 2000 :: NEXT I
240 CALL MAGNIFY (4)
250 GOTO 250
```

Nel programma precedente il codice di carattere specificato è 68 ("D"), che è divisibile per quattro. Lo sprite è formato, in sequenza, dalle lettere D E F G (codici 68, 69, 70, 71).
Per quanto detto prima, sostituendo nell'istruzione 220 il codice 68 con, ad esempio, 78 ("N") lo sprite sarebbe formato dalle lettere L M N O (codici 76, 77, 78, 79).

Risulta adesso chiaro che per sagome molto elaborate la parte più noiosa è quella relativa al calcolo degli identificatori esadecimali che devono ridefinire i caratteri. Fortunatamente esistono in commercio programmi chiamati Sprite Editor) che sollevano da questa incombenza rendendo anzi questa fase addirittura divertente. Nel prossimo numero di **Personal Software** presenteremo proprio un potente Sprite Editor, che potrete utilizzare per definire i vostri caratteri. Per ultimo diciamo che si può modificare la sagoma di uno sprite da programma, senza toccare le altre caratteristiche, con l'istruzione:
CALL PATTERN (# numero di sprite, codice di carattere)
dove il "numero di sprite" è quello dello sprite che vogliamo modificare ed il "codice di carattere", secondo quanto detto prima, è un intero tra 32 e 143 che si riferisce alla nuova sagoma. Provate ad aggiungere le seguenti linee al programma precedente ed osservate i risultati.

```
245 FOR I = 2 TO 2000 :: NEXT I
247 CALL PATTERN (# 2, 84)
```

Il colore del folletto

I colori disponibili sul TI 99/4A sono sedici, identificati da numeri interi a partire da uno. Di solito, di un insieme di caratteri si può sceglie-

re sia il colore del carattere (foreground) sia quello dello sfondo (background). Il colore standard dello sfondo per uno sprite è "trasparente" (cod. = 1) e non può essere modificato. Possiamo invece scegliere il colore del folletto attraverso il codice-colore appropriato nella CALL SPRITE e modificarlo in qualunque momento con l'istruzione:

CALL COLOR (# numero di sprite, codice-colore)

La posizione del folletto

Lo schermo del 99 è formato da 24 righe e 32 colonne. Dato che ogni carattere viene formato su una griglia di 8 x 8 pixel, l'intero schermo comprende 192 x 256 pixel. Quando ci riferiamo ad uno sprite, la posizione sullo schermo è riferita al pixel; il video è formato da 256 pirighe (di cui le prime 192 visibili) e da 256 pi-colonne. Le righe sono numerate, a partire da uno, dall'alto in basso, le colonne da destra a sinistra. Ad esempio, se vogliamo posizionare uno sprite sulla riga 15 e sulla colonna 6, dovremo specificare una pi-riga uguale a 113 ed una picolonna uguale a 41. La relazione è infatti:

pi-riga = (riga ★ 8) − 7

e ciò vale anche per le colonne. La posizione di partenza per un folletto è quella specificata nella CALL SPRITE, riferita all'angolo in alto a sinistra della sagoma che lo definisce.

La posizione può essere modificata con una istruzione del tipo:

CALL LOCATE (# numero di sprite, pi-riga, pi-colonna)

Il breve programma che segue muove a caso uno sprite sul video, mostrando le coordinate della posizione. Notate che se il numero di pi-riga generato è maggiore di 192, lo sprite non è visibile (fermate il programma con CLEAR).

```
200 CALL CLEAR :: RANDOMIZE
210 CALL MAGNIFY (3) :: CALL SPRITE (# 2, 80, 2, 200, 2)
220 RI = INT (RND ★ 256) + 1 :: CO = INT (RND ★ 256) + 1
230 DISPLAY AT (2, 1) SIZE (12) : "RIGA :"; RI
240 DISPLAY AT (2, 16) BEEP : "COLONNA :"; CO
250 CALL LOCATE (# 2, RI, CO)
260 FOR I = 2 TO 1500 :: NEXT I
270 GOTO 220
```

Il movimento del folletto

Il movimento di uno sprite è il risultato della somma vettoriale di due grandezze: la velocità di riga e la velocità di colonna. Esse sono rappresentate da numeri interi compresi tra − 128 e + 127. Se assumiamo l'origine di un piano cartesiano in alto a sinistra dello schermo, con l'asse delle ascisse orientato verso il basso e quello delle ordinate da sinistra verso destra, la velocità di riga è la componente della velocità secondo l'asse "x" e la velocità di colonna la componente secondo l'asse "y". In altre parole, una velocità di riga positiva muove lo sprite verso il basso, una velocità di colonna positiva muove lo sprite verso destra e viceversa.

Queste caratteristiche risultano utilissime quando si voglia ad esempio simulare, limitandosi al primo quadrante, il moto di un punto materiale soggetto a forze. L'istruzione per modificare la velocità di uno sprite è:

CALL MOTION (# numero di sprite, velocità di riga, velocità di colonna)

Il seguente programma muove a caso uno sprite, mostrando i valori relativi della velocità (fermare il programma con CLEAR):

```
200 CALL CLEAR :: RANDOMIZE
210 CALL SPRITE (# 2, 65, 2, 3, 3)
220 VR = INT (RND ★ 256) − 128 :: VC = INT (RND ★ 256) − 128
230 DISPLAY AT (2, 2) : "VELOCITA' DI RIGA :"; VR
240 DISPLAY AT (4, 2) BEEP : "VELOCITA' DI COLONNA :"; VC
250 CALL MOTION (# 2, VR, VC)
260 FOR I = 2 TO 2000 :: NEXT I
270 GOTO 220
```

Uno sprite si ferma con una istruzione del tipo:

CALL MOTION (# numero di sprite, 0, 0)

A velocità molto basse o molto alte lo sprite si muove in modo quasi perfetto. A velocità medie si nota qualche sobbalzo, ma il risultato è comunque ottimo. A velocità molto alte una sagoma perde qualcosa in definizione (è una questione ottica): in questi casi è bene scegliere per lo sprite un colore che sia ben visibile sullo sfondo.

Le informazioni sul folletto

L'Extended BASIC è dotato di una serie di istruzioni che permettono di controllare la "vita" di uno sprite. Vediamole un po' in dettaglio. L'istruzione:

CALL POSITION (# numero di sprite, pi-riga, pi-colonna)

serve a determinare la posizione di uno sprite nel preciso istante in cui viene eseguita. I valori ritornati nelle variabili numeriche "pi-riga" e "pi-colonna" sono, alla luce di quanto finora detto, numeri interi tra 1 e 256. Ad esempio l'istruzione:

CALL POSITION (# 2, RI, CO)

fa assumere alle variabili RI e CO i valori delle coordinate della posizione attuale dello sprite numero 2.

L'istruzione:

CALL COINC (# numero di sprite, pi-riga, pi-colonna, tolleranza, variabile numerica)

serve a verificare la sovrapposizione di uno sprite ad una posizione dello schermo. La posizione è quella identificata dalle variabili "pi-riga" e "pi-colonna". La "tolleranza" è il valore numerico dello scarto massimo consentito; il valore che assume la "variabile numerica" è zero se non c'è sovrapposizione, − 1 in caso

contrario. L'istruzione può anche essere utilizzata nelle forme:
a) CALL COINC (# numero di sprite, # numero di sprite, tolleranza, variabile numerica)
b) CALL COINC (ALL, variabile numerica)

Nella forma a) riporta la sovrapposizione di due sprite identificati dai rispettivi numeri. Nella forma b) riporta qualunque sovrapposizione di due o più sprite tra quelli utilizzati.
Una CALL COINC è attiva solo quando viene eseguita: nessuna sovrapposizione viene rilevata, se il computer sta eseguendo altre istruzioni. Ciò costringe a porre molta attenzione nel suo utilizzo, come vedremo in dettaglio nel seguito.

Infine l'istruzione:
CALL DISTANCE (# numero di sprite, # numero di sprite, variabile numerica)
ritorna nella "variabile numerica" il quadrato della distanza tra due sprite, calcolato applicando il ... teorema di Pitagora e riferito al pixel in alto a sinistra delle sagome. Questa istruzione può essere anche usata nella forma:
CALL DISTANCE (# numero di sprite, pi-riga, pi-colonna, variabile numerica)
in questo caso la distanza calcolata è quella fra lo sprite e la posizione dello schermo specificata dalle variabili "pi-riga" e "pi-colonna".

La morte del folletto

Uno sprite si cancella con l'istruzione:
CALL DEL SPRITE (# numero di sprite)
che può essere utilizzata anche nella forma:

CALL DEL SPRITE (ALL)

In quest'ultimo caso vengono cancellati tutti gli sprite usati in precedenza. Se il "numero di sprite" è quello di uno sprite non definito, il computer non segnala l'errore; questa caratteristica è utile in molti casi.

Note generali di utilizzo

Quanto detto finora può indurre a pensare che sia estremamente facile scrivere questo tipo di programmi. In effetti, le istruzioni a disposizione sono molte e ben "pensate", tali cioè da eliminare molti problemi. In realtà, per applicazioni in cui la velocità è essenziale le cose si complicano molto e a volte bisogna rinunciare a qualcosa per ottenere buoni risultati. Il difetto più grande è la relativa lentezza dell'interprete BASIC. Questo significa che già per velocità di grandezza intorno a 20 accade che:

• una CALL POSITION ritorna una posizione di qualche attimo indietro rispetto alla realtà;
• una sovrapposizione non viene rilevata, anche se esiste, se non per valori di tolleranza abbastanza alti;
• in caso di IF ... THEN, ogni istruzione eseguita in più fa perdere il sincronismo delle azioni.

Per questi motivi la stesura di programmi sufficientemente veloci costringe in genere a veri e propri salti mortali, come ad esempio:

• uso abbondante ed indiscriminato di GOTO, che tolgono in pratica ogni pretesa di eleganza e leggibilità al listato;
• ricorso ad espedienti per verificare le sovrapposizioni, perché una CALL COINC è praticamente inutilizzabile;
• riduzione all'osso degli effetti di contorno (ad esempio suoni, ecc.): ogni istruzione in più è un problema in più.

Il metodo probabilmente più efficace, dopo aver sviluppato l'algoritmo di movimento, è quello di affinare sul campo il programma, scegliendo alla fine i valori migliori per i parametri in gioco: velocità, posizione, tolleranze. Infatti, in pratica è impossibile sapere in anticipo, senza far girare il programma, cosa succederà sullo schermo. Il breve programma che segue è una esemplificazione degli argomenti trattati finora. Il listato è volutamente ridotto all'essenziale per permettere ad ognuno di voi di modificarlo a piacimento. Fate particolare attenzione alle tecniche utilizzate per ottenere certi risultati: con un po' di pratica vi accorgerete che tutto, entro certi limiti, può essere fatto.

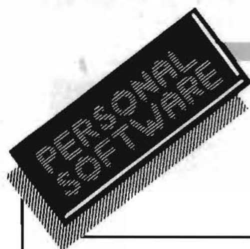
Darts

Il gioco è molto semplice: avete un "lanciadardi" che si muove sulla sinistra dello schermo dall'alto in basso e spara quando premete la barra spaziatrice. Bisogna abbattere un muro di 18 mattoni. Ad ogni passaggio il muro si allontana verso destra e la velocità del lanciadardi aumenta. Un mattone, per essere distrutto, deve essere colpito due volte. Vince chi abbatte il muro nel minor numero di passaggi. Anche se relativamente poco complesso, il gioco, nell'esecuzione, è abbastanza difficile. Diciotto passaggi è già un buon risultato e sfido chiunque a farcela in meno di quattrodici. La situazione del muro è memorizzata nel vettore MU (24). Le istruzioni di movimento degli sprite sono state a lungo testate per cercare il valore migliore per i parametri in gioco. Si è verificato, infatti, che una velocità del lanciadardi maggiore di 18 fa "saltare" la verifica dei passaggi. Infine la tecnica usata per visualizzare numeri o lettere di grandezza doppia può esservi utile in altri programmi.

Vi consiglio perciò di modificare il listato secondo i vostri gusti, magari aggiungendo una sigla ed un diverso sistema per il punteggio. Vi accorgerete che non è poi così difficile costruire da soli giochi anche molto complessi: basta un po' di pratica ed un buon "rodaggio" del programma.
Buon lavoro !!!!!

Commenti al programma Darts

200 - 210 Scelta del colore dello schermo (grigio). Definizione dei caratteri grafici secondo le seguenti corrispondenze: 129 = spazio; 130 =



Grafica avanzata sul TI 99/4A

mezzo mattone; 131 = mattone; 132 = lanciadardi; 133 = dardo.
220 Valori iniziali del muro, rappresentato dal vettore MU (24). In particolare il valore due significa "mattone intero", uno mezzo mattone, zero nessun mattone.
230 Valori iniziali delle variabili. K è il valore delle unità e KA quello delle decine del numero di passaggi effettuati. VE è la velocità del lanciadardi, COL è la posizione del muro.
240 - 260 La variabile J controlla la fine del gioco (J = 0).
Si stampa la situazione attuale del muro.
270 Lo sprite n. 2 è il lanciadardi. Si stampa il valore delle decine del numero dei passaggi (sprite n. 4).
280 Si stampa il valore delle unità del numero dei passaggi (sprite n. 5). Si imprime velocità al lanciadardi.
290 Verifica se è stata premuta la barra spaziatrice (codice ASCII 32).
300 In caso affermativo controlla la posizione del lanciadardi: se è troppo vicino al bordo inferiore dello schermo, non spara.
310 Crea e muove il dardo verso il muro. Calcola la posizione del muro da controllare.
320 Ciclo di ritardo variabile che permette al dardo di raggiungere il muro.
330 Controlla se un mattone è stato colpito ed aggiorna la situazione del muro.
340 Cancella il dardo.
350 Verifica se il lanciadardi ha raggiunto la base dello schermo.
360 In caso affermativo calcola i nuovi valori delle variabili ed il ciclo ricomincia. Notate che il valore logico "vero" sul 99 è rappresentato da − 1. Perciò la velocità del lanciadardi sarà al massimo 18 e la posizione del muro non supererà la colonna 28.
370 - 390 Se il gioco è finito, stampa il numero dei tentativi e chiede se si vuole continuare.
400 - 410 Ciclo di attesa della risposta. Se il tasto premuto è "S" (codice ASCII 83) si ricomincia.
420 Fine programma. ■

Othello-Reversi per VIC 20 e C 64

Se pensate che sia poco originale, leggete fino in fondo!

di *Alessandro Guida*

Il gioco

Othello, noto anche col nome di Reversi, è uno dei giochi che più si prestano all'implementazione su computer ed allo stesso tempo uno dei più affascinanti giochi da scacchiera.
Prima di analizzarne la realizzazione su VIC (senza espansioni) o C 64, due parole sul gioco per chi non lo conosce.
Due giocatori (Bianco e Nero) si fronteggiano su una scacchiera da 8 x 8. Entrambi hanno 32 pedine da un lato bianche e dall'altro nere. I giocatori usano le pedine dal lato del proprio colore.

In partenza, al centro della scacchiera (figura 1), ci sono 4 pedine (due per giocatore). A turno i giocatori pongono una pedina sulla scacchiera, a condizione che questa catturi almeno una pedina avversaria. Deve, cioè, effettuare una presa.
Per catturare una o più pedine dell'avversario, bisogna che queste siano comprese tra la nuova pedina ed una già presente sulla scacchiera ma dello stesso giocatore di turno.
Può accadere che in un solo turno si catturino più pedine in direzioni diverse, ma con in comune la pedina appena aggiunta sulla scacchiera.
Tutte le pedine "mangiate" vanno capovolte (cioè occorre cambiare il colore), e diventano del giocatore che ha effettuato la mossa. Dall'operazione di capovolgimento il gioco ha preso anche il nome di Reversi.

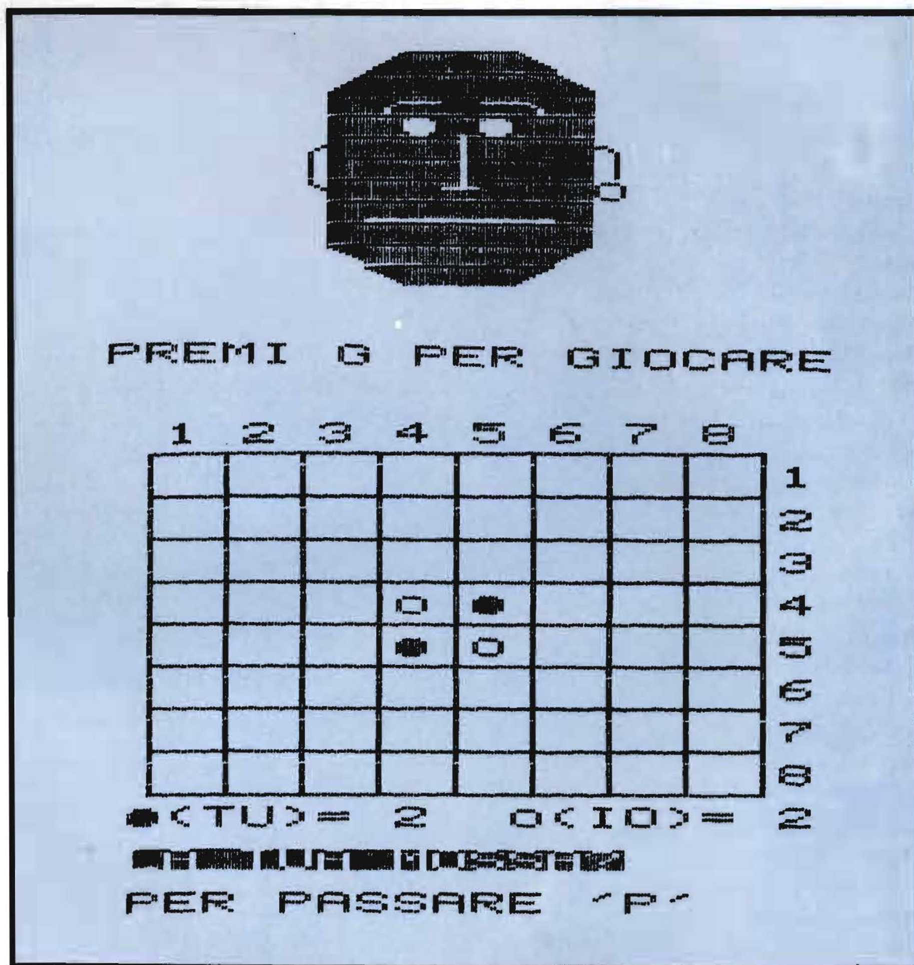


Figura 1. *Posizione iniziale della scacchiera per il gioco Othello.*

Un esempio di questa operazione è visualizzato nelle figure 2-3 e 4.
Se non è possibile effettuare una presa, si rinuncia al proprio turno.
La partita termina quando sono state depositate tutte le 64 pedine, o se uno dei due giocatori non ha più pedine del proprio colore sulla scacchiera o se entrambi i giocatori rinunciano alla mossa perchè non è possibile catturare alcuna pedina.

Implementazione dei giochi da scacchiera

Il flusso di un gioco classico (dama, scacchi, othello, ecc.) si può riassumere nei seguenti punti:
1) ricevi mossa dall'avversario;
2) controllane la validità ed eseguila;
3) cerca tutte le mosse possibili;
4) scegli quella migliore;
5) esegui la mossa scelta. Se la partita non è terminata torna al punto 1;
6) decreta il vincitore.

Tutte queste operazioni si prestano bene ad essere eseguite dal computer in quanto molto ripetitive. Ma nascono dei problemi al punto 4,

quando si tratta di scegliere la mossa migliore.
Evidentemente, la bravura di un giocatore sta nella sua capacità di scegliere in ogni momento la mossa migliore.

Perciò, per mettere il computer in grado di giocare in maniera non casuale, è necessario trovare un algoritmo in grado di valutare la situazione della scacchiera, e decidere se questa è favorevole o no.
Un algoritmo di tale genere è detto "funzione euristica".
Generalmente, però, un algoritmo, per quanto buono sia non garantisce la scelta della migliore mossa in assoluto.
Accade, infatti, che un momentaneo vantaggio si tramuti in seguito in una sconfitta, come avviene, ad esempio, spesso negli scacchi.

Per risolvere questo problema si fa ricorso alle strutture ad albero. Partendo dalla situazione della scacchiera si prendono in considerazione tutte le mosse possibili. Per ognuna di queste mosse si analizzano, poi, tutte le possibili mosse di risposta dell'avversario, e per ogni risposta tutte le ulteriori mosse possibili per il computer, e così via.
Sapremo così per ognuna delle mosse a cosa si va incontro in futuro. Inoltre, più si esplora l'albero delle possibili situazioni in profondità più sarà attendibile, alla fine, la scelta della mossa.

Purtroppo, questo sistema ci è precluso dalla lentezza del BASIC per cui dovremo concentrarci nella ricerca di una funzione euristica abbastanza valida.

La strategia di gioco

Per realizzare un algoritmo efficiente ho tenuto presente alcune regole che, dopo un paio di partite, appariranno molto importanti anche a voi.
1) La mossa è tanto migliore quante più sono le pedine catturate all'avversario.

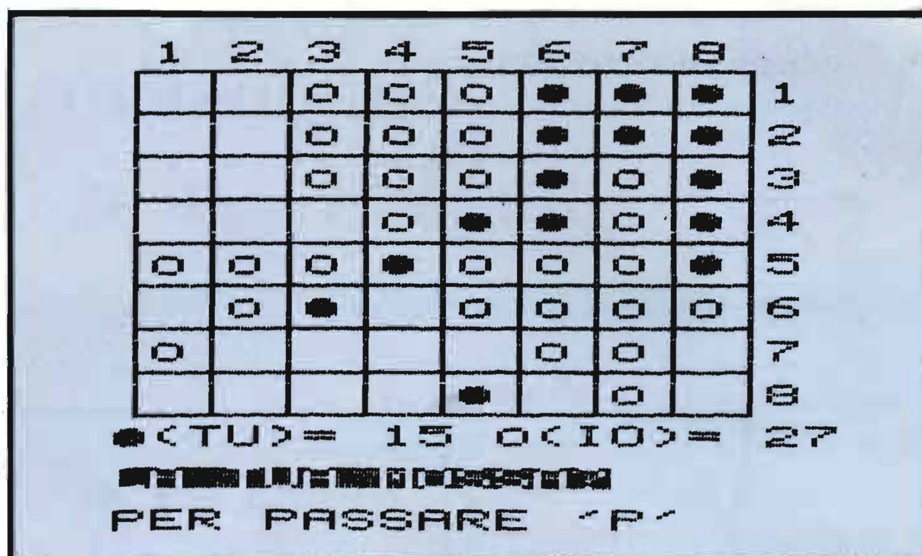


Figura 2. *Posizione della scacchiera prima della mossa dell'avversario.*

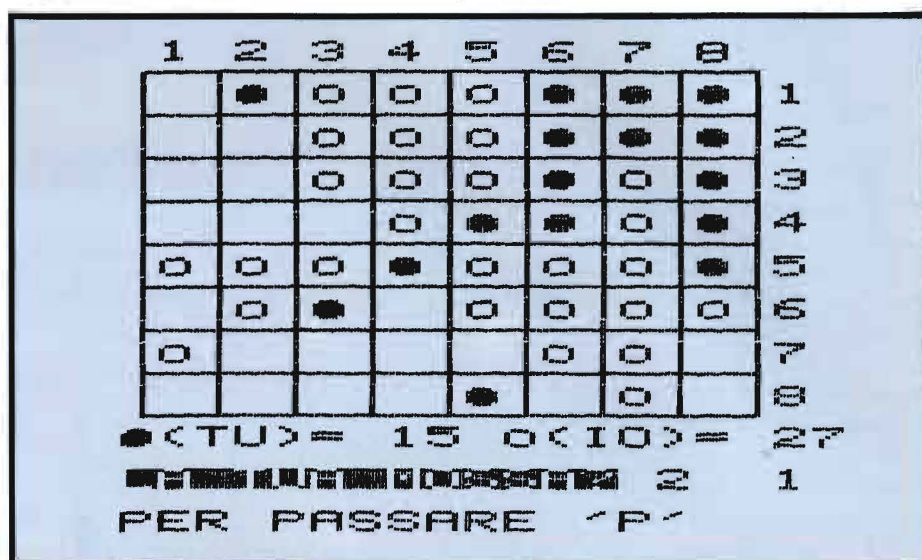


Figura 3. *L'avversario ha posto la sua pedina in 2, 1.*

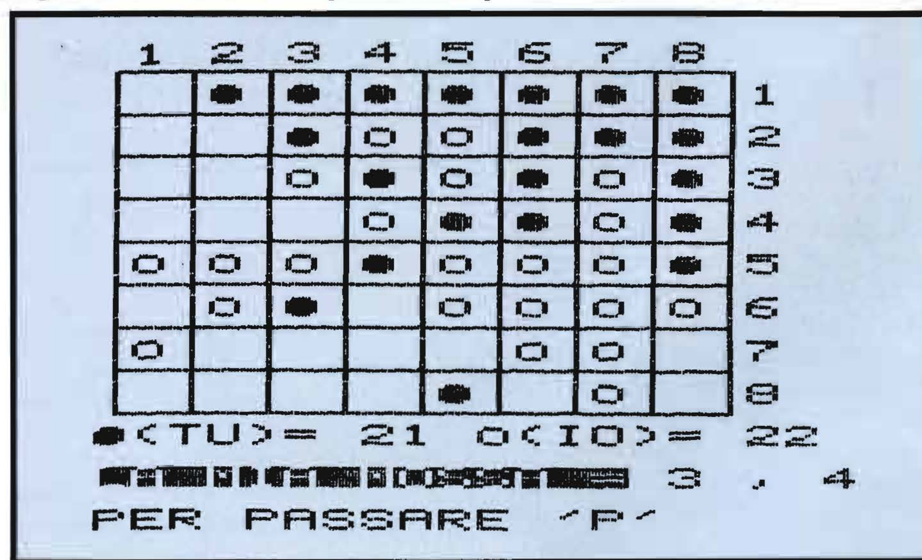


Figura 4. *Situazione della scacchiera dopo avere capovolto tutte le pedine catturate.*

2) La regola 1 non è sempre vera, poichè vi sono situazioni in cui è preferibile mangiarne meno ed occupare posizioni più sicure sulla scacchiera.
3) Infatti, una pedina posta al centro



Othello-Reversi per VIC 20 e C 64

della scacchiera sarà vittima di ripetuti capovolgimenti, in quanto facilmente attaccabile da tutte le direzioni. Al contrario le caselle ai vertici della scacchiera sono sicurissime perchè inattaccabili. Ecco i valori che ho assegnato alle caselle della scacchiera:

5	3	3	3	3	3	3	5
3	1	2	2	2	2	1	3
3	2	1	2	2	1	2	3
3	2	2	1	1	2	2	3
3	2	2	1	1	2	2	3
3	2	1	2	2	1	2	3
3	1	2	2	2	2	1	3
5	3	3	3	3	3	3	5

Per conoscere l'efficacia di una mossa basta sommare i valori delle pedine interessate. Più è alto il punteggio, migliore è la mossa.

4) Il punteggio calcolato al punto 3 va diminuito se in una direzione dopo la nostra pedina ce n'è una dell'avversario. Ad esempio in figura 2, serve a ben poco mettere una pedina nera in 2, 3 se, poi, il computer può mettere la sua bianca in 1,3 riprendendosi tutte le pedine toltegli al turno precedente.

Nella realizzazione pratica del programma ho trascurato il punto 4 per permettere al programma di girare anche sul VIC inespanso. È stata, invece, aggiunta la scelta casuale della mossa da eseguire nel caso di due o più mosse con lo stesso valore. In questa maniera si evita la ripetizione delle stesse situazioni.

Se avete abbastanza spazio in memoria potete provare ad aggiungere il controllo descritto al punto 4. Potete anche variare i valori assegnati alle caselle della scacchiera. Sicuramente, cambiando combinazioni il computer diventa più o meno difficile da sconfiggere.

Il programma

Per memorizzare la scacchiera è stato utilizzato un vettore di 72 elementi. Ogni riga è composta da 9

MODIFICHE PER IL C 64

Per far girare il programma Othello su di un C 64 è necessario modificare le linee 1, 32, 34, 52 e 94 come segue:

```
1 R=RND(-TI):POKE54296,15:WS=54272:POKEWS+5,9:
POKEWS+6,0:GOTO109
32 FORI=0TO50:POKEWS+0,20:POKEWS+1,117:POKEWS+4,33:
NEXT:GOTO109
52 POKEWS+0,81:POKEWS+1,7:POKEWS+4,33
94 POKEWS+0,20:POKEWS+4,33:FORM=0TO7:FORN=1TO8
```

Poco prima di consegnare questo articolo, ho provato a far giocare due computer ad Othello. In uno (A) ho introdotto il programma del listato 1.
Nell'altro (B) ho caricato lo stesso programma ma ho cambiato i data in:

```
105   DATA   9,5,5,5
106   DATA   5,1,3,3
107   DATA   5,3,1,3
108   DATA   5,3,3,1
```

Su 6 partite B ne ha vinte 4, 1 è finita pari, e l'altra l'ha vinta A. Conviene quindi usare quest'ultimo gruppo di data al posto di quelli che appaiono nel listato 1.
Provate anche voi a mettere il vostro computer contro quello di un vostro amico, ognuno con i suoi "data segreti".

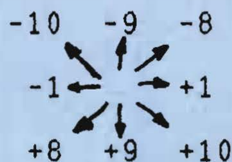


Tabella 1. *Valori da sommare al numero dell'elemento del vettore per spostarsi da una casella all'altra.*

```
----   --------------------------
!  0!  !  1!  2!  3!  4!  5!  6!  7!  8!
----   --------------------------
!  9!  !10!11!12!13!14!15!16!17!
----   --------------------------
!   !  !                        !

!   !  !                        !
----   --------------------------
!63!   !64!65!66!67!68!69!70!71!
----   --------------------------
```

Tabella 2. *Corrispondenza tra gli elementi del vettore e le caselle della scacchiera. La prima colonna contiene gli elementi che indicano il limite della scacchiera.*

Listato 1. *Il programma Othello versione VIC 20.*

```
1 R=RND(-TI):POKE36878,15:GOTO109
2 DIMS%(71),R%(8),K%(71)
3 FORCL=1TO8:READQ:NEXT
4 FORR=0TO3:FORCL=1TO4
5 N=R*9+CL:READQ:K%(N)=Q:K%(72-N)=Q
6 N=(7-R)*9+CL:K%(N)=Q:N=R*9+9-CL:K%(N)=Q
7 NEXTCL,R
8 S%(31)=-1:S%(41)=-1:S%(32)=-2:S%(40)=-2:PC=2:PA=2
9 I$="[illegible]"

11 FORI=0TO7:S%(I*9)=1:NEXT
12 PRINT"JINIZI TU?(S/N)"
13 GETA$:IFA$=""THEN13
14 PRINT"J":GOSUB92
15 IFA$="S"THEN35
16 IFPA/2>=PCTHENPRINTI$:PRINT"■* SONO NEI GUAI! *■":GOTO19
17 IFPC/2>=PATHENPRINTI$:PRINT"■MODESTAMENTE VINCO■":GOTO19
18 PRINTI$:PRINT"■* STO PENSANDO!  *■"
19 GOSUB57
20 IFC<>0THENMO=MO+1:K=C:G=-1:JP=0:GOTO25
21 IFJP=1THEN29
22 PRINTI$
23 PRINT"■PASSO■                  "
24 FORI=1TO1500:NEXT:GOTO35
25 PRINTI$
26 PRINT"■LA MIA MOSSA =■    ■■■";K-INT(K/9)*9;",";INT(K/9)+1
27 GOSUB73
28 IFMO<60ANDPA>0THEN35
```

elementi contigui del vettore; il primo elemento contiene 1 per indicare il limite della scacchiera (tabella 2). In questo modo, per passare da una casella ad un'altra è sufficiente sommare uno dei valori riportati in tabella 1.

Nel vettore conserveremo: 0 = per indicare una casella vuota; 1 = fuori scacchiera; −1 = pedine computer; −2 = pedine avversario.

Ecco un breve commento alle linee del programma che è presentato nel listato 1 nella versione per VIC 20. Le modifiche per il C 64 sono riportate nel riquadro.

2 Dimensiona vettori scacchiera (S%), valore caselle (K%), direzioni (R%).

3-7 Costruisce il vettore valore caselle partendo dai valori di un solo quadrante della scacchiera.

8 Posiziona le prime quattro pedine.

11 Memorizza il limite della scacchiera.

12-15 Inizio partita con scelta del primo giocatore.

16-27 Gioca il computer.

28 Controlla se è terminata la partita.

29-34 Se la partita è terminata decreta il vincitore.

35-50 Riceve la mossa dell'avversario la controlla e la esegue.

51-56 Messaggi di errore.

57-72 Routine scelta mossa del computer.

57-60 Loop principale per la ricerca delle caselle vuote.

61-72 Routine per il calcolo del valore della mossa e la scelta della migliore.

73-80 Routine controllo mosse. Se T = 0 la mossa non effettua prese.

81-88 Routine esecuzione mossa, aggiornamento vettore scacchiera.

88-91 Conteggio numero pedine computer ed avversario.

92-103 Routine disegno scacchiera e punteggi.

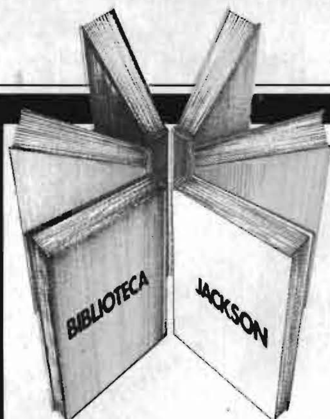
104 Data direzioni movimento sulla scacchiera.

105-108 Data valori caselle del secondo quadrante della scacchiera.

109-127 Routine copertina. ■

Seguito listato 1.

```
29 PRINTI$
30 IFPC<PATHEN33
31 PRINT"■HO VINTO■                    "
32 FORI=254TO128STEP-1:POKE36875,I:FORP=1TO20:NEXT:NEXT:POKE36875,0:GOTO109
33 PRINT"■HAI VINTO■                   "
34 FORI=128TO254:POKE36875,I:FORP=1TO20:NEXT:NEXT:POKE36875,0:GOTO109
35 PRINTI$
36 PRINT"■LA TUA MOSSA?■          ■■■■■■■■■";
37 GETA$:IFA$=""THEN37
38 IFA$="P"THENPRINTI$:PRINT"                                 ":JP=1:GOTO16
39 IFVAL(A$)<1ORVAL(A$)>8THEN37
40 CL=VAL(A$):PRINTCL;
41 GETA$:IFA$=""THEN41
42 IFVAL(A$)<1ORVAL(A$)>8THEN41
43 R=VAL(A$)-1:PRINTR+1
44 K=CL+R*9:ER=0
45 IFS%(K)<>0THENER=2:GOTO51
46 G=-2:GOSUB73
47 IFT=0THENER=3:GOTO51
48 MO=MO+1:PRINTI$:PRINT"                    "
49 IFMO<60ANDPC>0THEN16
50 GOTO29
51 PRINTI$
52 POKE36874,150
53 IFER=2THENPRINT"CASELLA OCCUPATA      ":GOTO55
54 PRINT"NON REALIZZA PRESA   "
55 FORI=1TO1500:NEXT:POKE36874,0
56 PRINT"        ":FORI=1TO1500:NEXT:GOTO35
57 P=0:C=0
58 FORK=1TO71
59 IFS%(K)=0THENGOSUB61
60 NEXT:RETURN
61 PM=0:PP=0:RESTORE:FORD=1TO8
62 READQ:H=K:F=0:N=0
63 H=H+Q
64 IFH<0ORH>71THEN68
65 IFS%(H)=0ORS%(H)=1THEN68
66 IFS%(H)=-2THENF=F+K%(H):N=N+1:GOTO63
67 IFH<>K+QTHENF=F+K%(H):PM=PM+F:PP=PP+N
68 NEXT:IFPP=PATHENP=1E4:C=K:RETURN
69 IFPP=0THENRETURN
70 PM=PM+K%(K):IFPM<PTHENRETURN
71 IFPM=PANDRND(1)>.5THENRETURN
72 P=PM:C=K:RETURN
73 RESTORE:T=0
74 FORI=1TO8:R%(I)=0:READQ
75 H=K
76 H=H+Q
77 IFH>71ORH<0THEN80
78 IFS%(H)=(-3-G)THEN76
79 IFS%(H)=GANDS%(H-Q)=(-3-G)THENR%(I)=Q:T=1
80 NEXT:IFT=0THENRETURN
81 S%(K)=G
82 GOSUB92
83 FORI=1TO8:IFR%(I)=0THEN88
84 H=K
85 H=H+R%(I)
86 IFS%(H)=(-3-G)THENS%(H)=G:GOTO85
87 GOSUB92
88 NEXT:PC=0:PA=0
89 FORI=1TO71:IFS%(I)=-1THENPC=PC+1
90 IFS%(I)=-2THENPA=PA+1
91 NEXT
92 PRINT"■ 1 2 3 4 5 6 7 8"
93 PRINT" ┌┬┬┬┬┬┬┬┐"
94 FORM=0TO7:POKE36876,-223*(M<2):FORN=1TO8
95 IFS%(N+M*9)=0THENPRINT"| ";:GOTO98
96 IFS%(N+M*9)=-1THENPRINT"|O";:GOTO98
97 PRINT"|●";
98 NEXT:PRINTM+1"■■■|";
99 PRINT:PRINT" ├┼┼┼┼┼┼┼┤"
100 NEXTM:PRINT"■└┴┴┴┴┴┴┴┘"
101 PRINT"●(TU)=    ■■■"PA;TAB(10)"O(IO)=    ■■■"PC
102 PRINT"■■■PER PASSARE 'P'";
103 RETURN
104 DATA-10,-9,-8,-1,1,8,9,10
105 DATA5,3,3,3
106 DATA3,1,2,2
107 DATA3,2,1,2
108 DATA3,2,2,1
109 PRINT"■":B$="■    ■":B1$="      "
110 FORJ=1TO1000:NEXT:PRINT"■■■"
111 PRINTTAB(7)" ■     ■"
112 PRINTTAB(7)"■       ■"
113 PRINTTAB(7)"■ ╭ ╮  ■"
114 PRINTTAB(7)"■  ● ●  ■"
115 PRINTTAB(6)"/■   |   ■\"
116 PRINTTAB(6)"|■   |   ■|"
117 PRINTTAB(6)"\■   ┴   ■O"
118 PRINTTAB(7)"■ ■";B$;"■ ■"
119 PRINTTAB(7)"■ ■";B1$;"■ ■"
120 PRINTTAB(7)"■■      ■"
121 PRINTTAB(7)" ■■    ■"
122 PRINT"■■■";TAB(7)N$
123 PRINT"■■■■PREMI G PER GIOCARE■";
124 GETR$:IFR$<>"G"THEN126
125 CLR:GOTO2
126 IFB$="■_____■"THENB$="■    ■":B1$="       ":N$="■OTHELLO■":GOTO110
127 B$="■_____■":B1$="■     ■":N$="         ":GOTO110
```



Libri firmati JACKSON



VOI E IL VOSTRO COMMODORE 64
Un esaudiente vademecum sulla programmazione in BASIC dal Personal ad oggi tra i più diffusi.
Facile, brillante ricco di programmi verificati, questo è un prezioso volume sia per i neofiti che per gli utilizzatori più esigenti.
256 pag. **L. 22.000**
Cod. **347 B.**

PROGRAMMAZIONE DELLO ZX SPECTRUM
Aggiungete suono e colore ai vostri programmi, scoprite lo SPECTRUM negli affari e nell'istruzione, giocate e imparate a scrivere i giochi, disegnate figure in 3 dimensioni.
212 pag. **L. 18.000**
Cod. **531 D.**

APPLE MEMO
Sintassi dei comandi, codici dei caratteri, messaggi di errore, linguaggio macchina, indirizzi utili.....
Un libro destinato a stare in permanenza a fianco del vostro Apple.
146 pag. **L. 15.000**
Cod. **340 H.**

BASIC SU APPLE
programmi in pochi minuti
65 programmi pronti che vi risolveranno problemi che vanno dalla "economia domestica", alle applicazioni commerciali, ai calcoli statistici, alla creazione degli archivi.
184 pag. **L. 14.000**
Cod. **532 H.**

APPLE TUTTO FARE
Collegamenti e progetti
Questo libro è stato scritto per chiunque voglia capire come l'APPLE e gli altri home computer, possano essere interfacciati con il mondo esterno.
208 pag. **L. 18.000**
Cod. **334 D.**

PET/CBM GUIDA ALL'USO VOL. 1 e VOL. 2
È la versione italiana del famosissimo testo americano: "PET/CBM Personal Computer Guide" ed è presentato in due volumi data l'ampiezza e la profondità degli argomenti trattati. In questo manuale troverete tutto ciò che è necessario sapere sui calcolatori COMMODORE.
VOL 1 250 pag. **L. 20.000**
cod. **332 P**
VOL 2 282 pag. **L. 22.500**
cod. **333 P**

La Biblioteca che fa testo



GRUPPO EDITORIALE JACKSON

Attenzione compilare per intero la cedola
ritagliare (o fotocopiare) e spedire in busta chiusa a:
GRUPPO EDITORIALE JACKSON
Divisione Libri
Via Rosellini, 12 - 20124 Milano

CEDOLA DI COMMISSIONE LIBRARIA

VOGLIATE SPEDIRMI

n° copie	codice	Prezzo unitario	Prezzo totale
		Totale	

☐ Pagherò contrassegno al postino il prezzo indicato più **L. 2000 per contributo fisso spese di spedizione**

Condizioni di pagamento con esenzione del contributo spese di spedizione:

☐ Allego assegno della Banca ____________
n° ____________

☐ Allego fotocopia del versamento su c/c n. 11666203 a voi intestato

☐ Allego fotocopia di versamento su vaglia postale a voi intestato

Nome
Cognome
Via
Cap Città Prov.
Data Firma

Spazio riservato alle Aziende. Si richiede l'emissione di fattura
Partita I.V.A. | | | | | | | | | | |

ORDINE MINIMO L. 50.000



Routine in linguaggio macchina per ZX81

— Parte seconda —

Alcune interessanti subroutine da inserire nei vostri programmi

di *Carlo Cappelli*

Chi ha realizzato i programmi presentati nell'articolo precedente si sarà reso conto delle incredibili possibilità che si possono ottenere con la programmazione in linguaggio macchina.

Lo scopo di questo secondo articolo è sia di approfondire i campi di applicazione dei programmi presentati precedentemente, che di sviluppare nuove idee di utilità generale.

È anche opportuno fare delle precisazioni sulla differenza tra RAND USR ... e LET K = USR ... per chiamare le routine; bisogna infatti tener presente che l'istruzione RAND stabilisce il punto di partenza della sequenza dei numeri casuali ottenibili con la funzione RND ed è quindi sconsigliabile usare RAND USR ..., se nel programma si usa anche la funzione RND, altrimenti si rischia di avere sempre lo stesso numero casuale o la stessa sequenza di numeri casuali.

Come primo argomento viene perfezionata la routine per congiungere due punti presentata nella scorsa puntata; a tale scopo abbiamo ancora una ventina di byte a disposizione nella REM.

Vengono presentati tre miglioramenti del modo di caricare i punti da congiungere. Per inserire il miglioramento scelto occorre caricare la versione veloce del programma di unione dei punti (1 REM ..., 5 REM ...) e successivamente scrivere il programma di caricamento codici, presentato all'inizio dell'articolo precedente, però con:

10 LET A = 16720

Fatto ciò, passiamo ad analizzare i programmi.

Con il primo miglioramento le coordinate del punto di arrivo si caricano nella stringa W$, mentre le coordinate del punto di partenza sono quelle dell'ultimo punto disegnato con il comando PLOT. Quindi, per tracciare una linea dal punto (X, Y) al punto (A, B) si scrive:

```
PLOT X, Y
LET W$ = CHR$ A + CHR$ B
RAND USR 16720
```

Questo è possibile grazie al fatto che le coordinate dell'ultimo punto disegnato da PLOT vengono memorizzate nelle locazioni 16438 e 16439.

Per implementare il miglioramento occorre modificare la routine di caricamento delle coordinate, che incomincia dalla locazione 16720, inserendo i codici decimali della figura 1.

La differenza con il caricatore precedente consiste anche in un controllo più preciso per localizzare la stringa W$. Infatti viene cercato, oltre al codice di W$ (che è 92), anche il valore della lunghezza che, essendo di due caratteri, in memoria è scritta 2,0. Così facendo si cerca di evitare il più possibile la confusione con una serie numerica simile, casualmente incontrata.

Per rendere operativo il programma, dopo aver inserito i codici, occorre introdurre il listato 1.

Con questo programmino si unisce il punto 1,1 con il punto 21,21 (il codice del + = 21); si noti che si può definire prima la stringa e poi eseguire il PLOT e viceversa, l'importante è che entrambe le operazioni avvengano prima della chiamata della routine.

Faccio notare che il programma non presenta controlli, quindi la stringa W$ *deve essere* di 2 caratteri,

16720	LD HL, (16396)	42	12	64
	LD A, (HL)	126		
	INC HL	35		
	CP 92	254	92	
	JR NZ − 5	32	250	
	LD A, (HL)	126		
	CP 2	254	2	
	JR NZ − 10	32	245	
	INC HL	35		
	LD A, (HL)	126		
	AND A	167		
	JR NZ − 15	32	240	
	INC HL	35		
	LD A, (HL)	126		
	LD (16516).A	50	132	64
	INC HL	35		
	LD A, (HL)	126		
	LD (16517).A	50	133	64
	LD HL, (16438)	42	54	64
	LD (16514).HL	34	130	64
16755	JP 16532	195	148	64

Figura 1. *Listato Assembly e codici decimali. Per il caricamento delle coordinate dei punti da plottare vedere dump 1.*

Routine in linguaggio macchina per ZX81

i cui valori devono risultare appropriati (i codici corrispondenti ai caratteri non devono superare il primo 63 e il secondo 43). Le applicazioni sono molteplici, ma principalmente può servire per prospettive di funzioni, cioè per proiettare in un unico punto (quello plottato inizialmente) le varie posizioni di un grafico, come nel programma del listato 2.

Per vedere i disegni così composti bisogna prima dare il RUN, eseguire poi GOTO 110 e poi ancora GOTO 210.

Il secondo miglioramento permette di inserire in una unica stringa una serie di punti (al massimo 126) da unire due a due.

Per inserire questo miglioramento bisogna compiere le stesse operazioni descritte per il primo, cioè caricare il programma base (le due REM) e le istruzioni per il caricamento dei programmi in linguaggio macchina, con 10 LET A = 16720; dare il RUN e inserire i codici della figura 2.

Per usare questa routine si scrive: POKE 16731, LEN W$ RAND. USR 16720.

Ad esempio, se si vogliono disegnare tre segmenti di coordinate: 10, 10 − 23, 17 per il primo; 0, 0 − 7, 9 per il secondo; 63, 0 − 0, 43 per il terzo, come dal listato 3.

Con il programma del listato 4 si possono realizzare tutte le assonometrie di un cubo e, con semplici modifiche, lo si può adattare per visualizzare altri solidi. La figura 8 illustra il passaggio dalle coordinate spaziali a quelle sul piano.

Al primo input bisogna inserire un valore fino a 20; al secondo e terzo il punto intorno al quale si vuol visualizzare il cubo (esempio centro schermo 31 − 21); al quarto e al quinto input l'angolo assonometrico (esempio, per l'assonometria cavaliera inserire: 45 e 0).

Il terzo miglioramento serve per congiungere più punti consecutivi: è adatto quindi a rappresentare le figure sul piano.

È da notare che con i programmi di disegno migliorati si può sopperire alla mancanza di istruzioni come SQUARE ecc.; infatti si può definire la stringa W$ con le coordinate di qualsiasi struttura geometrica, in modo che la routine disegni sullo schermo la figura voluta.

Per inserire quest'ultimo miglioramento bisogna procedere come nei casi precedenti e inserire i codici della figura 3.

Per tracciare, ad esempio, due segmenti di coordinate: (0.0) - (30,30) per il primo e (30,30) - (3,43) per il secondo bisogna inserire il listato 5.

Si possono realizzare composizioni grafiche gradevoli, come nel listato 6, oppure si possono emulare funzioni grafiche come "square".

Sempre sfruttando questo miglioramento è stato realizzato un programma per la rotazione di figure sul piano, presentato nel listato 7.

In input va inserito il numero dei lati della figura da ruotare (che deve essere inferiore a 126) e le coordinate dei vertici di ogni segmento che compone la figura, ricordando che le suddette coordinate non sono riferite all'origine in basso a sinistra, ma al centro dello schermo; quindi per far ruotare un quadrato rispetto al centro delle sue diagonali si dovrà

16720	LD HL. (16396)	42	12	64
	LD A. (HL)	126		
	INC HL	35		
	CP 92	254	92	
	JR NZ − 5	32	250	
	LD A. (HL)	126		
	CP 0	254	0	
	JR NZ − 10	32	245	
	INC HL	35		
	INC HL	35		
	RRA	31		
	RRA	31		
	LD B. A	71		
	PUSH BC	197		
	LD BC. 4	1	4	0
	LD DE. 16514	17	130	64
	LDIR	237	176	
	PUSH HL	229		
	CALL 16532	205	148	64
	POP HL	225		
	POP BC	193		
	DJNZ − 16	16	239	
16756	RET	201		

Figura 2. *Listato Assembly e codici decimali del secondo miglioramento.*

16720	LD HL (16400)	42	16	64
	LD A. (HL)	126		
	INC HL	35		
	CP 92	254	92	
	JR NZ − 5	32	250	
	LD A, (HL)	126		
	CP 0	254	0	
	JR NZ − 10	32	245	
	INC HL	35		
	INC HL	35		
	RRA	31		
	DEC A	61		
	LD B. A	71		
	PUSH BC	197		
	LD BC 4	1	4	0
	LD DE. 16514	17	130	64
	LDIR	237	176	
	PUSH HL	229		
	CALL 16532	205	148	64
	POP HL	225		
	POP BC	193		
	DEC HL	43		
	DEC HL	43		
	DJNZ − 18	16	237	
16758	RET	201		

Figura 3. *Listato Assembly e codici decimali del terzo miglioramento.*

PERSONAL SOFTWARE

Routine in linguaggio macchina per ZX81

inserire:

X1 = − 5	Y1 = − 5
X2 = − 5	Y2 = 5
X3 = 5	Y3 = 5
X4 = 5	Y4 = − 5
X5 = − 5	Y5 = − 5

Per le figure chiuse, come nel caso di un quadrato, la quinta coordinata deve essere uguale alla prima; questo perché il programma non distingue tra figure chiuse e aperte.
Per quanto riguarda i parametri estremi delle coordinate vale la regola: $x^2 + y^2 < 441$. In linea di massima si possono inserire valori di x o di y compresi tra − 21 e 21, purché rispettino la regola appena esposta; comunque ci pensa il programma a verificare se i parametri sono esatti.
Per fare una rotazione continua della figura basta inserire:

450 GOTO 340

Invece se si vuole fare in modo che la figura lasci una traccia mentre ruota, basta togliere le linee 400 e 410.

Per ottenere immagini insolite o rotazioni diverse si possono modificare gli argomenti trigonometrici alla linea 290.
La 290 si può scrivere così:
290 LET A$ = (C, 2 x U − I TO) = CHR$ (A (U, I) x COS (A (U, 2) x C + G) + 31) + CHR$ (A (U, I) x SIN (A (U, 2) x C + G) + 21).
Provate ora a inserire il quadrato precedente (meglio se al posto dei 5 inserite i 10, poiché la figura risulta più grossa) dopo aver tolto le linee 400 e 410: otterrete una immagine simile a un fiore.
Oppure al posto di A (U, 2) x C fate A (U, 2)/C e vedrete il quadrato ridursi.
A titolo di suggerimento, potete provare a modificare, sempre nella linea 290, la variabile G inserendo: A (U, 2) ★ G x C oppure ... A (U, 2) + G/C ...
La rotazione delle figure avviene di trenta gradi in trenta gradi; per cambiare questo valore bisogna modificare lo step alla linea 270 e cambiare opportunamente il valore 13 nelle linee 260 e 340 nel modo seguente: nuovo valore = 2 x PI / nuovo step + 1.
Ad esempio, per una rotazione di venti gradi in venti gradi, con uno step quindi di PI/9, il valore sostitutivo deve essere di 19.

Con questi miglioramenti si è potuto vedere come, con semplici modifiche, si possono adattare dei programmi in linguaggio macchina a esigenze particolari; ora passiamo ad analizzare nuove e interessanti idee per potenziare lo ZX81; non si dimentichi che queste stesse routine possono essere adattate anche per altri personal che usino il microprocessore Z 80.

Il prossimo programma serve per imitare i comandi che gestiscono i colori (nel nostro caso solo per il bianco e per il nero) posseduti da molti BASIC. In sostanza, il programma serve per invertire il colore di una parte di schermo definita dall'utente.
Per questo programma occorre una REM di almeno 90 caratteri e il programma di caricamento della routine in linguaggio macchina presentato all'inizio dell'articolo precedente.
I codici da inserire sono quelli della figura 4.
Prima di usare questa routine biso-

16518	NOP	0		
	LD HL. (16400)	42	16	64
	LD A. (HL)	126		
	INC HL	35		
	CP 92	254	92	
	JR NZ − 5	32	250	
	LD A. (HL)	126		
	CP 4	254	4	
	JR NZ − 10	32	245	
	INC HL	35		
	INC HL	35		
	LD DE. 16514	17	130	64
	LD BC. 4	1	4	0
	LDIR	237	176	
	LD HL; (D-FILE)	42	12	64
	INC HL	35		
	LD D. 0	22	0	
	LD A. (16515)	58	131	64
	LD E. A	95		
	LD B. 33	6	33	
	ADD HL. DE	25		
	DJNZ − 2	16	253	
	LD A; (16514)	58	130	64
	LD E. A	95		
	ADD HL. DE	25		
	LD A. (16517)	58	133	64
	LD (16518). A	50	134	64
	LD A. (16516)	58	132	64
	LD B. A	71		
	LD A. (HL)	126		
	ADD A. 128	198	128	
	LD (HL). A	119		
	INC HL	35		
	DJNZ − 6	16	249	
	LD A. (16516)	58	132	64
	LD E. A	95		
	LD A. 33	62	33	
	SUB E	147		
	LD E. A	95		
	ADD HL. DE	25		
	LD A. (16518)	58	134	64
	DEC A	61		
	RET Z	200		
	LD (16518). A	50	134	64
16597	JR − 29	24	226	

Figura 4. *Listato Assembly e codici decimali della routine per l'inversione di zone video.*

gna inserire i parametri della finestra di schermo da invertire nella stringa W$ secondo la solita regola, cioè prima le coordinate del punto estremo in alto a sinistra, poi la lunghezza e infine la larghezza.
Per invertire un rettangolo di schermo, il cui estremo in alto a sinistra ha coordinate 8,8, di lunghezza 15 e larghezza 7, si deve scrivere la stringa in questo modo:
W$ = CHR$ 8 + CHR$ 8 + CHR$ 15 + CHR$ 7.
I valori da inserire devono essere congrui con le dimensioni dello schermo, come più volte precisato, altrimenti si rischia di bloccare il computer.
Per la chiamata alla routine si può usare: RAND USR 16519. Si provi il programmino del listato 8.
La routine presentata è utile per far lampeggiare messaggi o immagini.

Sempre restando nel tema di gestire immagini video, come terzo argomento viene potenziato il programma per inviare istantaneamente immagini sullo schermo (programma presentato nell'articolo precedente). Le modifiche comportano un nuovo modo di memorizzare l'immagine video, che rende possibile anche il suo trasferimento su cassetta, e la gestione veloce dei fotogrammi che, attraverso un'apposita routine, permette animazioni spettacolari.
Cominciamo, quindi, col caricare il programma dell'articolo precedente, controllando che la REM abbia 75 caratteri. Se avete seguito le indicazioni precedenti, non avrete alcun problema, altrimenti modificate la REM e, per essere sicuri che sia della lunghezza giusta, scrivete PRINT PEEK 16594; dovete ottenere 234.
Successivamente dovete comporre una seconda REM (esempio, 5 REM) di almeno 85 caratteri. I codici vanno inseriti con il solito programma di caricamento, però con: 10 LET A = 16795 (vedi figura 5).
parti distinte; la prima, che iniza dalla locazione 16595, serve per trovare nell'area variabili la stringa A$,

16595	LD HL. (16516)	42	132	64
	LD D. 0	22	0	
	LD B. L	69		
	LD E. H	92		
	LD HL 0	33	0	0
	ADD HL. DE	25		
	DJNZ – 2	16	253	
	LD B. 0	6	0	
	EX DE. HL	235		
	LD HL. 0	33	0	0
	LD A. B	120		
	ADD HL. DE	25		
	DJNZ – 2	16	253	
	EX DE. HL	235		
	LD (16656). A	50	16	65
	LD HL. (16400)	42	16	64
	LD A. H	124		
	CP 125	254	125	
	JR Z 20	40	20	
	LD A. (HL)	126		
	INC HL	35		
	CP 70	254	70	
	JR NZ – 10	32	245	
	LD A. (HL)	126		
	CP E	187		
	JR NZ – 9	32	246	
	INC HL	35		
	LD A. (HL)	126		
	CP D	186		
	JR NZ – 14	32	241	
	INC HL	35		
	LD (16547). HL	34	163	64
	RET	201		
	LD A. 16	62	16	
	LD (16384). A	50	0	64
	RET	201		
16656	NOP	0		
16657	CALL 16519	205	135	64
	LD (16547). HL	34	163	64
	LD A. (16656)	58	16	65
	DEC A	61		
	JP Z 16595	202	211	64
	LD (16656). A	50	16	65
16673	RET	201		

Figura 5. *Listato Assembly e codici decimali della modifica per il trasferimento delle immagini video.*

che contiene tutti i dati dei disegni; la seconda parte gestisce l'alternarsi dei fotogrammi. Questo ciclo, però, non è chiuso in linguaggio macchina, ma vi è un ritorno al BASIC per poter regolare la velocità con la quale si devono alternare i fotogrammi. Chi volesse ottenere la massima velocità, chiudendo il ciclo in linguaggio macchina, deve sostituire l'ultima istruzione (16673 RET) con: JR – 18 24 237, anche se la cosa non è molto consigliabile, causa l'eccessiva velocità ottenuta.

Per attivare questa routine bisogna, oltre a inserire i soliti parametri dimensionali descritti nell'articolo precedente, scrivere nella locazione

Routine in linguaggio macchina per ZX81

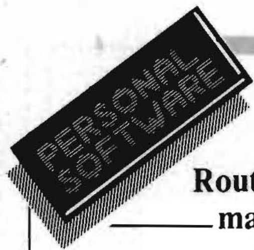
16609 il numero dei riquadri preparati; per conoscere il numero massimo di questi si ricordi che non si possono superare le disponibilità di memoria dei computer, quindi con l'espansione da 16 Kbyte si possono inserire al massimo circa 22 riquadri di 600 caratteri l'uno (30 x 20).
Successivamente si deve usare RAND USR 16595 per identificare la posizione della stringa A$ e, dopo RAND USR 16657 si devono inviare le immagini sullo schermo. Le applicazioni di questo programma sono per tutte le simulazioni di movimento; cercate ad esempio di imitare l'effetto tunnel, preparando pochi fotogrammi che diano l'idea di avanzare dentro una galleria.
Se volete sbizzarrire la fantasia aggiungete alle due REM il programma del listato 9, che permette di preparare i vari fotogrammi.

Per un corretto funzionamento del programma si consiglia di non inserire dimensioni eccessive, cioè di non comporre disegni grandi come tutto lo schermo, poiché si rischia di coprire i messaggi che il programma BASIC invia sullo schermo stesso. Non preoccupatevi se nel comporre il disegno commettete degli errori poiché, alla fine di ogni riquadro, il programma vi chiede se dovete correggere qualche linea.
Se volete realizzare il già accennato effetto tunnel, è sufficiente inserire le immagini 3, 45.
Per centrare opportunamente la figura bisogna inserire come X di partenza il valore 6 e per la Y il valore 1.
Chi ha fantasia e inventiva avrà già capito i molteplici impieghi possibili di questa routine nella realizzazione di giochi di movimento.
Il prossimo argomento che affronteremo è quello della protezione dei programmi contro accessi non autorizzati, imitando una caratteristica dei grandi sistemi.

La routine presentata può essere inserita all'inizio di qualsiasi programma e ne blocca l'esecuzione fino a quando non viene inserita correttamente la parola chiave, che de-

16550	codici	Esempio	A	38		
	parola		B	39		
	chiave		C	40		
	di 5		D	41		
	caratteri		E	42		
16555	PUSH HL			229		
	PUSH DE			213		
	CALL 699			205	187	2
	LD A. H			124		
	ADD A. 2			198	2	
	JR C 8			56	8	
	LD B. H			68		
	LD C. L			77		
	CALL 1981			205	189	7
	LD C. (HL)			78		
	JR C 2			56	2	
	LD C. 0			14	0	
	POP DE			209		
	POP HL			225		
	RET			201		
16578	LD HL. (D-FILE)			42	12	64
	INC HL			35		
	LD (HL). 146			54	146	
	LD D. 0			22	0	
	CALL 16555			205	171	64
	LD A. C			121		
	AND A			167		
	JR Z − 6			40	249	
	DEC A			61		
	CP 117			254	117	
	JR Z 23			40	23	
	INC D			20		
	LD A. D			122		
	CP 20			254	20	
	JR Z 11			40	11	
	LD (HL). C			113		
	INC HL			35		
	CALL 16555			205	171	64
	LD A. C			121		
	AND A			167		
	JR NZ − 6			32	246	
	JR − 28			24	227	
	CALL 2602			205	42	10
	JP 16578			195	194	64
	LD A. D			122		
	CP 5			254	5	
	JR NZ − 10			32	245	
	LD HL. (D-FILE)			42	12	64
	LD DE. 16550			17	166	64
	LD B. 5			6	5	
	INC HL			35		
	LD C. (HL)			78		
	EX DE. HL			235		
	LD A. (HL)			126		
	CP C			185		
	JR NZ − 25			32	230	
	INC HL			35		
	EX DE. HL			235		
	DJNZ − 10			16	245	
	CALL 2602			205	42	10
16648	RET			201		

Figura 6. *Listato Assembly e codici decimali della routine per la protezione dei programmi mediante chiave di accesso.*



Routine in linguaggio macchina per ZX81

ve essere composta da 5 lettere o numeri.
Tutte le diverse chiavi possibili con 36 caratteri (26 lettere + 10 numeri), presi 5 a 5, sono ben 60.466.176.

La routine di protezione, che conviene inserire e memorizzare da sola per poterla richiamare prima di scrivere il programma che si vuole proteggere, presenta le seguenti caratteristiche:
1) è autolanciante, cioè parte automaticamente appena caricata. Per avere questa caratteristica è necessario che il comando "SAVE", usato per memorizzare il programma da proteggere, sia contenuto all'interno della routine, prima della prima istruzione che si vuole sia eseguita. Per memorizzare il programma in modo autolanciante basterà quindi eseguire "GOTO" seguito dal numero di linea dove si trova il comando "SAVE", mentre memorizzando il programma con il comando "SAVE" dato in modo immediato non si ottiene la partenza automatica;
2) appena partita entra in una routine in linguaggio macchina che non può essere fermata in nessun modo, a meno che non si vada a resettare la CPU con un intervento hardware;
3) la routine in linguaggio macchina accetta i caratteri premuti sulla tastiera, visualizzandoli sullo schermo e, ogni volta che si preme NEW LINE, verifica se i caratteri scritti sul video corrispondono alla parola chiave: in questo caso vi è il ritorno al BASIC.
Per inserire la routine scrivete innanzitutto: 1 SAVE "nome programma".
Il nome del programma deve essere al massimo di 10 lettere ed una volta scritto non deve essere più né allungato né accorciato, altrimenti si scompigliano le locazioni di memoria successive.
Poi scrivete 5 RAND USR 16578
10 REM ... con 120 caratteri
come dal listato 10.
A questo punto scrivete il solito programmino di caricamento, però con 11 LET A = 16550; fatto questo, inserite i codici della figura 6.
La routine in linguaggio macchina presenta delle chiamate a routine della ROM; per la precisione le CALL 699 e 1981 servono per l'analisi dei tasti premuti, mentre la CALL 2602 serve per pulire lo schermo. Se si vuol chiamare la parola chiave, non conviene editare la REM e correggerla, ma inserire, mediante il comando POKE, i nuovi codici nelle locazioni 16550-1-2-3-5.

Vediamo ora come è possibile superare questo tipo di protezione; infatti questo programma, come tutti i programmi autolancianti, può essere intercettato con il seguente programmino in linguaggio macchina, inseribile in una REM:

CALL 3875	205	35	15
SCF	55		
JP 835	195	67	3

Per attivare questa semplice routine basta dare il comando RAND USR 16514, dopo di che accendere il registratore. Il trucco consiste nel creare nella locazione 16384 un codice di errore, in modo tale che il computer, dopo avere seguito il LOAD si blocca, poiché va ad analizzare quella locazione di memoria. Per migliorare la protezione anche contro questo intervento, oppure nel caso che qualcuno riesca a bloccare il programma, si può usare un artificio; si può infatti spostare il puntatore di fine programma BASIC alla linea 5, in modo tale da non rendere visibile tutto ciò che si trova oltre quella linea, così chi tentasse di inserire delle istruzioni per cercare di decodificare ciò che si trova oltre la linea 5 non farebbe altro che cancellare il programma nascosto. L'unica possibilità è di ripristinare, a colpo sicuro, il puntatore di fine programma.

Per mettere in pratica anche questa protezione occorre:
1) che il nome del programma sia di 6 lettere;
2) inserire il programma di protezione, con l'aggiunta di alcune istruzioni. Dalla locazione 16648, al posto di RET inserire:

16648			
LD HL. 2560	33	0	10
LD (16541). HL	34	157	64
RET	201		

3) prima di salvare il programma su nastro eseguire POKE 16541, 118 e POKE 16542, 118: dopo questi due comandi non sarà più visibile tutto ciò che si trova oltre la linea 5;
4) salvare il programma su nastro con GOTO 1 e verificare che solo inserendo la parola chiave si può accedere al resto del programma.
Faccio notare che la procedura di spostamento del puntatore BASIC può essere utile nel caso si debba cancellare una serie di istruzioni terminali: basta "pokare", all'inizio della prima istruzione da cancellare, un 118.
L'ultimo argomento presentato consiste nell'implementazione sullo ZX81 uno SCROLL bidirezionale di porzioni di schermo definite dall'utente. Il programma, sfruttato opportunamente, può generare contemporaneamente parti di schermo che scorrono verso l'alto e parti che scorrono verso il basso oppure lo scroll di porzioni di schermo non quadrate o rettangolari, ma di varie forme; si riescono, quindi, a superare le limitazioni imposte dagli scroll standard.

Per incominciare scrivete una REM di 210 caratteri e con il solito programmino inserite i codici della figura 7 a partire dalla locazione 16518:

Figura 7. *Listato Assembly e codici decimali del programma di scroll bidirezionale verticale.*

16518	LD A. 16	62	16	
	LD (16383). A	50	0	64
	RET	201		
	LD HL. (16400)	42	16	64
	LD A. H	124		
	CP 125	254	125	
	JR Z – 13	40	242	
	LD A. (HL)	126		
	INC HL	35		
	CP 92	254	92	

Seguito figura 7.

	JR NZ – 10	32	245	
	LD A. (HL)	126		
	CP 6	254	6	
	JR NZ – 10	32	245	
	LD BC. 4	1	4	0
	INC HL	35		
	INC HL	35		
	LD DE. 16514	17	130	64
	LDIR	237	176	
	LD A. (HL)	126		
	CP 55	254	55	
	JR Z 4	40	4	
	LD A. 0	62	0	
	JR 2	24	2	
	LD A. 7	62	7	
	LD (16658). A	50	18	65
	INC HL	35		
	LD A. (HL)	126		
	CP 39	254	39	
	JR Z 11	40	11	
	LD HL. 6400	33	0	25
	LD (16636). HL	34	252	64
	LD A. (16515)	58	131	64
	JR 14	24	14	
	LD HL. 21229	33	237	82
	LD (16636). HL	34	252	64
	LD A. (16515)	58	131	64
	LD E. A	95		
	LD A. (16517)	58	133	64
	ADD A. E	131		
	LD HL. (D-FILE)	42	12	64
	INC HL	35		
	LD D. 0	22	0	
	LD E. A	95		
	LD B. 33	6	33	
	ADD HL. DE	25		
	DJNZ – 2	16	253	
	LD A. (16514)	58	130	64
	LD E. A	95		
	ADD HL. DE	25		
	LD A. (16517)	58	133	64
	LD (16514). A	50	130	64
	PUSH HL	229		
	LD A. (16516)	58	132	64
	LD C. A	79		
	LD DE. 16680	17	40	65
	LDIR	237	176	
	POP HL	225		
	PUSH HL	229		
	LD DE. 33	17	33	0
	SBC HL. DE	237	82	
	POP DE	209		
	LD A. (16516)	58	132	64
	LD C. A	79		
	LDIR	237	176	
	LD C.A	79		
	SBC HL. BC	237	66	
	LD A. (16514)	58	130	64
	DEC A	61		
	LD (16514). A	50	130	64
	JR NZ – 24	32	231	
	JR 0	24	0	
	LD B. C	65		
	LD (HL). 0	54	0	
	INC HL	35		
	DJNZ – 4	16	251	
	RET	201		
	LD DE. 16680	17	40	65
	EX DE. HL	235		
	LDIR	237	176	
16672	RET	201		

I tipi di scroll possibili sono 4: verso il basso; verso l'alto; verso il basso con rotazione (l'ultima linea viene stampata al posto della prima che scrolla verso il basso); verso l'alto con rotazione. I codici relativi ai diversi scroll sono: "SB"; "SA"; "RB"; "RA". Le dimensioni e il tipo di scroll devono essere inserite nella stringa W$ così: prima le coordinate del vertice in alto a sinistra, poi le dimensioni con altezza diminuita di uno e infine il codice dello scroll.
Ad esempio, per scrollare un rettangolo di schermo di dimensioni 16 x 7 e con il vertice in alto a sinistra di coordinate 3,3 occorre scrivere:
W$ = CHR$ 3 + CHR$ 3 + CHR$ 16 + CHR$ 6 + "SA"
così facendo lo scroll sarà verso l'alto (SA). La chiamata alla routine la si effettua con: RAND USR 16524.
Nel caso la stringa non abbia tutti e 6 i caratteri il programma si fermerà con errore H. Si veda il listato 11.
Se si vuole far scorrere una porzione di schermo fatta a L, la possiamo scomporre in due rettangoli per i quali definiremo lo scroll. Esempio:
1) rettangolo di vertice 6,1 e dimensioni 7 x 19;
2) rettangolo di vertice 13,13 e dimensioni 9 x 7.
Si deve inserire:

```
10 LET W$ = CHR$ 6 + CHR$ 1 + CHR$ 7 + CHR$ 18 + "SB"
20 RAND USER 16524
30 LET W$ = CHR$ 13 + CHR$ 13 + CHR$ 9 + CHR$ 6 + "SB"
40 RAND USR 16524
```

Con questo semplice programmino e la routine in linguaggio macchina si otterrà l'effetto desiderato; cioè lo scroll verso il basso di una porzione di schermo fatta a L.

Provate a scrollare in direzioni opposte due porzioni di schermo che si intersecano; otterrete risultati interessanti.

■

Routine in linguaggio macchina per ZX81

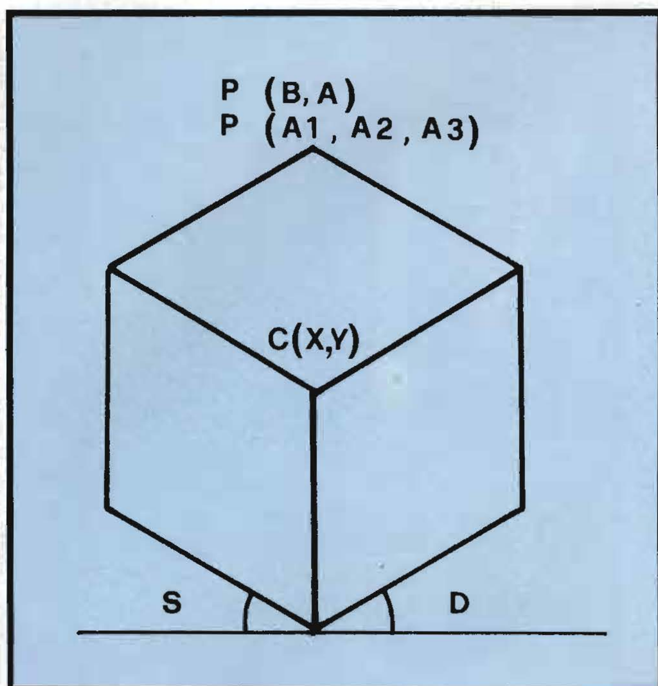


Figura 8. *Il passaggio tra le coordinate spaziali P e quelle sul piano P', avviene tramite le seguenti formule assonometriche: B = X − A1 x cosS + A2 x cosD e A = Y + A1 x sinS + A2 x sinD + A3.*

```
10 POKE 16803,128
20 FOR G=0 TO 42 STEP 2
30 PLOT 0,0
40 LET W$=CHR$ 63+CHR$ G
50 RAND USR 16720
60 PLOT 63,42
70 LET W$=CHR$ 0+CHR$ (42-G)
80 RAND USR 16720
90 NEXT G
100 STOP
110 FOR G=0 TO 42 STEP 2
120 PLOT 0,21
130 LET W$=CHR$ 63+CHR$ G
140 RAND USR 16720
150 PLOT 63,21
160 LET W$=CHR$ 0+CHR$ (42-G)
170 RAND USR 16720
180 NEXT G
200 STOP
210 PLOT 31,21
220 FOR G=0 TO 63 STEP 2
230 LET W$=CHR$ G+CHR$ 42
240 RAND USR 16720
250 LET W$=CHR$ (63-G)+CHR$ 0
260 RAND USR 16720
270 IF G>42 THEN GOTO 320
280 LET W$=CHR$ 63+CHR$ G
290 RAND USR 16720
300 LET W$=CHR$ 0+CHR$ (42-G)
310 RAND USR 16720
320 NEXT G
```

Listato 1. *Esempio di utilizzo del primo miglioramento, che traccia una linea sul video. Nella linea 10 si inserisca il valore 128 per plottare, 64 per non plottare.*

```
  1 REM  ++MBINKEY$MUINKEY$TAN
 MIINKEY$MINKEY$INKEY$TAN U RND?
U RND ?K"Y  ?YXLN  RND/: RETURN
 4 LN  RND/ YWLN  RNDU RND?U RND
 ?K"Y  ?YXLN  RND/: RETURN  4 LN
  RND/ YWLN  RND? S ??Y MUINKEY$
/ Y MINKEY$INKEY$?M RNDM3INKEY$M
.INKEY$-    K UNPLOT  ?M/INKEY$
?M-INKEY$E RND6(INKEY$   LN  INK
EY$2=Y  RETURN =S-2= M/INKEY$U(
INKEY$XM(INKEY$U) INKEY$XM) INKEY$
/)M/INKEY$U(INKEY$ M(INKEY$U) INK
EY$XM) INKEY$U RNDXCOS M RND??INK
EY$E£RNDM RND7RND6 RND? RNDCOS /
 DIM  ..........................
.......
   5 REM Y? ??YF ?: RETURN  S £
£CHR$   INKEY$?? RETURN  S £CHR
$  - ?E£RND;U INKEY$?? S; ( CLEAR
 6 INKEY$  ??52PI; ( CLEAR ?  U I
NKEY$ RETURN RNDC57TAN RETURN  S
 /  E INKEY$?? RETURN  "" ?TAN
?( RUN TAN RETURN  S   /<=
...................
  10 POKE 16803,128
  15 LET W$="++"
  20 PLOT 1,1
  30 RAND USR 16720
```

Listato 2. *Esempio pratico di utilizzo del primo miglioramento, che disegna tre differenti immagini grafiche utilizzando diversi algoritmi di tracciamento.*

```
  1 REM  2 FHBINKEY$MUINKEY$TAN
 MIINKEY$MINKEY$INKEY$TAN U RND?
U RND ?K"Y  ?YXLN  RND/: RETURN
 4 LN  RND/ YWLN  RNDU RND?U RND
 ?K"Y  ?YXLN  RND/: RETURN  4 LN
  RND/ YWLN  RND? S ??Y MUINKEY$
/ Y MINKEY$INKEY$?M RNDM3INKEY$M
.INKEY$-    K UNPLOT  ?M/INKEY$
?M-INKEY$E RND6(INKEY$  22LN  INK
EY$2$Y  RETURN .S-2. M/INKEY$U(
INKEY$WM(INKEY$U) INKEY$XM) INKEY$
/)M/INKEY$U(INKEY$WM(INKEY$U) INK
EY$ M) INKEY$U RNDXCOS M RND??INK
EY$E(RND73X?VAL   ) RND GOSUB
 FAST LN  RND LPRINT AT FF( GOSU
B TAN ..........................
..........
   5 REM  ? ??YF ?: RETURN  S £
£CHR$   INKEY$?? RETURN  S £CHR
$  - ?E£RND;U INKEY$?? S; ( CLEAR
 6 INKEY$  ??52PI; ( CLEAR ?  U I
NKEY$ RETURN RNDC57TAN RETURN  S
 /  E INKEY$?? RETURN  "" ?TAN
?( RUN TAN RETURN  S   /<=
...................
  10 POKE 16803,128
  20 LET W$=CHR$ 0+CHR$ 0+CHR$ 3
0+CHR$ 30+CHR$ 3+CHR$ 43
  30 POKE 16731,LEN W$
  40 RAND USR 16720
```

Listato 3. *Esempio di utilizzo del secondo miglioramento descritto nel testo.*

Routine in linguaggio macchina per ZX81

```
  10 LET A$="0001001001011010010
0100000001001011011010011011111
1011110110110010110 10"
  14 POKE 16731,48
  15 LET W$=""
  16 PRINT "****CUBI IN ASSONOM
ETRIA****"
  17 PRINT ,,
  20 PRINT "RAGGIO SFERA CIRCOSC
RITTA ";
  25 INPUT R
  27 PRINT R
  30 PRINT "CENTRO ASSE X ";
  35 INPUT X
  37 PRINT X
  40 PRINT "CENTRO ASSE Y ";
  45 INPUT Y
  47 PRINT Y
  50 PRINT "ANGOLO ASSONOMETRICO
 SINISTRO"
  52 PRINT "IN GRADI  ";
  55 INPUT AS
  57 PRINT AS
  60 PRINT "ANGOLO ASSONOMETRICO
 DESTRO"
  62 PRINT "IN GRADI  ";
  65 INPUT AD
  67 PRINT AD
  70 LET AS=AS*PI/180
  72 LET CS=COS AS
  73 LET SS=SIN AS
  75 LET AD=AD*PI/180
  77 LET CD=COS AD
  78 LET SD=SIN AD
  79 LET D=R/SQR 3
  80 FOR G=1 TO 72 STEP 3
  90 LET A1=D*((A$(G)="1")-(A$(G
)="0"))
 100 LET A2=D*((A$(G+1)="1")-(A$
(G+1)="0"))
 110 LET A3=D*((A$(G+2)="1")-(A$
(G+2)="0"))
 120 LET B=X-A1*CS+A2*CD
 130 LET A=Y+A1*SS+A2*SD+A3
 140 IF B*A<0 OR B>63 OR A>43 TH
EN GOTO 500
 145 LET A=A-A*(A<0 OR A>43)+43*
(A>43)
 150 LET W$=W$+CHR$ B+CHR$ A
 160 NEXT G
 170 PRINT
 180 PRINT "IN CAMPO INVERSO ?"
 190 INPUT A$
 195 CLS
 200 IF A$(1)="S" THEN GOTO 300
 210 POKE 16803,128
 220 RAND USR 16720
 230 STOP
 300 FOR G=1 TO 22
 310 PRINT "
"
 320 NEXT G
 330 POKE 16803,64
 340 GOTO 220
 500 PRINT "ERRORE DI DIMENSIONA
MENTO"
 510 STOP
```

Listato 4. *Programma per realizzare cubi in assonometria utilizzando il secondo miglioramento. Questo programma BASIC deve essere usato insieme alla parte in linguaggio macchina del listato 3.*

```
   1 REM    FHBINKEY$MUINKEY$TAN
 MIINKEY$MINKEY$INKEY$TAN U RND?
U RND ?K"Y  ?YXLN  RND/:  RETURN
 4 LN  RND/ YULN  RNDU RND?U RND
 ?K"Y  ?YXLN  RND/:  RETURN  4 LN
  RND/ YULN  RND? S ??Y  MUINKEY$
/ Y  MINKEY$INKEY$?M RNDM3INKEY$M
.INKEY$-     X  UNPLOT   ?M/INKEY$
?M-INKEY$E RND6(INKEY$  Z LN  INK
EY$2FY3  RETURN  ZS-2Z M/INKEY$U(
INKEY$WM(INKEY$U) INKEY$XM) INKEY$
/)M/INKEY$U(INKEY$WM(INKEY$U) INK
EY$ M) INKEY$U RNDXCOS M RND??INK
EY$E£RND?33?VAL    ) RND GOSUB
 FAST LN  RND LPRINT AT ( LOAD T
AN ...............................
..........
   5 REM REM ? ??YF ?:   RETURN
S £ECHR$   INKEY$??  RETURN  S E
CHR$  - ?E£RND;U INKEY$?? 5; ( CL
EAR 6 INKEY$   ??62PI; ( CLEAR ?
U INKEY$ RETURN RNDC57TAN RETURN
 5  /   INKEY$?? RETURN  ""  ?T
AN 7( RUN TAN RETURN  S   /  /(=
..........................
  10 POKE 16803,128
  20 LET W$=CHR$ 10+CHR$ 10+CHR$
 23+CHR$ 17+CHR$ 0+CHR$ 0+CHR$ 7
+CHR$ 9+CHR$ 63+CHR$ 0+CHR$ 0+CH
R$ 43
  30 POKE 16731,LEN W$
  40 RAND USR 16720
```

Listato 5. *Esempio di utilizzo del terzo miglioramento contenuto nelle REM iniziali.*

```
  10 POKE 16731,10
  20 POKE 16803,128
  22 FOR T=1 TO 2
  25 FOR U=2 TO 5
  30 FOR I=U TO 40 STEP 5
  40 LET X=20-I/2
  50 LET Y=X
  60 LET W$=CHR$ X+CHR$ Y+CHR$ (
X+I)+CHR$ Y+CHR$ (X+I)+CHR$ (Y+I
)+CHR$ X+CHR$ (Y+I)+CHR$ X+CHR$
Y
  70 RAND USR 16720
  80 NEXT I
  90 NEXT U
 100 POKE 16803,64
 110 NEXT T
 120 GOTO 20
```

Listato 6. *Composizione grafica di quadrati sfruttando il terzo miglioramento. Questo programma deve essere usato insieme alla routine in linguaggio macchina contenuta nelle REM del listato 5.*

Listato 7. *Programma BASIC per realizzare la rotazione di figure piane con la routine in linguaggio macchina del terzo miglioramento. Questo programma deve essere usato in unione alla routine contenuta nelle REM del listato 5.*

```
  10 PRINT "*********ROTAZIONI**
*******"
  20 PRINT ,,,
  30 PRINT "IL CENTRO DI ROTAZIO
NE E' IL CEN  TRO DELLO SCHERMO (
31,21)"
  40 PRINT
```

Routine in linguaggio macchina per ZX81

Seguito listato 7.

```
  50 PRINT "NUMERO LATI DELLA FI
GURA ";
  60 INPUT L
  70 CLS
  80 POKE 16731,2*(L+1)
  90 DIM A(L+1,2)
 100 PRINT "INSERISCI ";L+1;" SP
IGOLI DELLA  FIGURA RISPETTO AL
CENTRO"
 110 PRINT ,,
 120 FOR G=1 TO L+1
 130 PRINT "X";G;"=";
 140 INPUT X
 150 PRINT X,"Y";G;"=";
 160 INPUT Y
 170 PRINT Y
 200 LET A(G,1)=SQR (Y*Y+X*X)*(S
GN X+1*(X=0))
 210 IF ABS A(G,1)>21 THEN STOP
 220 IF X=0 THEN LET A(G,2)=PI/2
*SGN Y
 230 IF X=0 THEN GOTO 250
 240 LET A(G,2)=ATN (Y/X)
 250 NEXT G
 255 FAST
 260 DIM A$(13,2*(L+1))
 265 LET C=1
 270 FOR G=0 TO 2*PI STEP PI/6
 280 FOR U=1 TO L+1
 290 LET A$(C,2*U-1 TO )=CHR$ (A
(U,1)*COS (A(U,2)+G)+31)+CHR$ (A
(U,1)*SIN (A(U,2)+G)+21)
 300 NEXT U
 310 LET C=C+1
 320 NEXT G
 325 SLOW
 330 CLS
 340 FOR G=1 TO 13
 350 POKE 16803,128
 360 LET W$=A$(G)
 370 RAND USR 16720
 380 FOR U=1 TO 10
 390 NEXT U
 400 POKE 16803,64
 410 RAND USR 16720
 420 NEXT G
 430 POKE 16803,128
 440 RAND USR 16720
```

```
   1 REM ▓▓?▛ E(RND7!▙RND▘,   GOS
UB ▙E£RND7- U▖RND?▘5; ( CLEAR U▙R
ND?;U▙RNDM▘RNDU▙RND? RAND U▜RND?
Y5▇?;U▘RNDXCOS M▘RND/ LLIST ....
................................
................................
   3 FOR G=1 TO 22
   4 PRINT "LKIO██ █ ███████LLLLJ
HHYYYYJGDFFF"
   5 NEXT G
  10 LET W$=CHR$ 8+CHR$ 8+CHR$ 1
5+CHR$ 7
  20 RAND USR 16519
```

Listato 8. *Esempio di utilizzo dell'invertitore di caratteri.*

Listato 9. *Programma per la composizione di immagini e per la realizzazione di animazioni.*

```
   1 REM ▓▙▌▘E£RND7- U▖RND?▘5; (
CLEAR U▙RND?; FOR U▌RNDM▘RND5 R
ETURN ?U▜RND? GOSUB █ FAST STR$
?Y5- LPRINT ▓?; FOR LPRINT U▘RN
DXCOS M▘RND/ STOP ...........
   5 REM E▜RND- ??5  ; ( CLEAR ▞
 FOR 6  ?; ( CLEAR FOR M(INKEY$E(
RND? RETURN ?C=█4 LET 76█RNDTAN
Y(M RNDTAN  LN ▖RND6█RNDU(INKEY$
XASN PEEK RNDM(INKEY$TAN .......
................................
................................
...........
```

Seguito listato 9.

```
  10 PRINT "LARGHEZZA DISEGNO ";
  20 INPUT LX
  25 PRINT LX
  30 POKE 16516,LX
  40 PRINT "LUNGHEZZA DISEGNO ";
  50 INPUT LY
  55 PRINT LY
  60 POKE 16517,LY
  80 PRINT "X DI PARTENZA ";
  90 INPUT X
  95 PRINT X
 100 POKE 16514,X
 110 PRINT "Y DI PARTENZA ";
 120 INPUT Y
 125 PRINT Y
 130 POKE 16515,Y
 140 PRINT "NUMERO IMMAGINI ";
 150 INPUT N
 155 PRINT N
 160 POKE 16609,N
 170 PRINT ,,
 180 PRINT "COMPONI I DISEGNI LI
NEA PER LINEA"
 190 PAUSE 1000
 195 CLS
 200 LET A$=""
 210 DIM B$(LX)
 220 FOR G=1 TO N
 230 PRINT AT 0,0;G
 240 PRINT AT Y,X;
 250 FOR U=1 TO LY
 260 INPUT B$
 270 LET A$=A$+B$
 280 PRINT TAB X;B$
 290 NEXT U
 300 PRINT AT 0,3;"DEVI CORREGGE
RE ?"
 310 INPUT Q$
 320 IF Q$="S" OR Q$="SI" THEN G
OSUB 1000
 330 CLS
 340 NEXT G
 350 RAND USR 16595
 360 FOR G=1 TO N
 370 RAND USR 16657
 380 FOR U=1 TO 6
 390 NEXT U
 400 NEXT G
 410 STOP
1000 CLS
1010 LET K=(G-1)*LY*LX+1
1020 FOR J=0 TO LY-1
1030 PRINT J+1;")";TAB 3;A$(K+J*
LX TO K+(J+1)*LX-1)
1040 NEXT J
1050 PRINT AT 21,0;"QUALE LINEA
(PER FINIRE BATTI 0)"
1060 INPUT J
1070 IF J=0 THEN RETURN
1080 PRINT AT 21,0;"RISCRIVI LA
LINEA"
1090 INPUT B$
1100 LET A$(K+LX*(J-1) TO K+LX*J
-1)=B$
1110 PRINT AT J-1,3;B$
1120 GOTO 1050
```

```
   1 SAVE "CODIC█"
   5 RAND USR 16578
  10 REM ....ABCDE FAST STR$ LN
█▘?LEN ▘5▓??LN █▀?5▘: SGN LPRINT
 TAN E£RND?0█- LN █RND?█C RAND X
 RETURN ?C*=? RETURN =C"??LN █RN
D?█4 RAND / STOP LN E▀?TAB RND?
RETURN ▌4 PRINT E£RND)█RND▞▌?? F
OR ( PRINT LN E▀TAN ...........
```

Listato 10. *Routine per la protezione dei programmi da accessi non autorizzati.*

Routine in linguaggio macchina per ZX81

Dump 1. *Dump di controllo utile per inserire e verificare i codici del listato Assembly della figura 1. Si tratta di una modifica alla routine di tracciamento punti presentata nella puntata precedente.*

1)

```
16720 = 42
16721 = 12
16722 = 64
16723 = 125
16724 = [illegible]
16725 = 254
16726 = 92
16727 = 32
16728 = 250
16729 = 126
16730 = 254
16731 = 2
16732 = 32
16733 = 245
[illegible]
16735 = 128
16736 = 167
16737 = 32
16738 = 240
16739 = 35
16740 = 126
16741 = 50
16742 = 132
16743 = 64
16744 = 35
[illegible]
16746 = 50
16747 = 133
16748 = 64
16749 = 42
16750 = 54
16751 = 64
16752 = 34
16753 = 130
16754 = 64
16755 = 195
16756 = 146
16757 = 64
```

2)

```
16720 = 42
16721 = 12
16722 = 64
16723 = 126
16724 = 35
16725 = 254
16726 = 92
16727 = 32
16728 = 250
16729 = 126
16730 = 254
16731 = 12
16732 = 32
16733 = 246
16734 = 35
16735 = 35
16736 = 31
16737 = 31
16738 = 71
16739 = 197
16740 = 1
16741 = 4
16742 = 0
16743 = 17
16744 = 130
16745 = 64
16746 = 237
16747 = 176
16748 = 229
16749 = 205
16750 = 148
16751 = 64
16752 = 225
16753 = 193
16754 = 16
16755 = 239
16756 = 201
```

Dump 2. *Dump di controllo della routine in linguaggio macchina della figura 2.*

```
   1 REM ▀(▘Y(M RNDTAN E(RND? R
ETURN ?C PAUSE ▪ 77)▙RND GOSUB ▩
 / ▘Y▛M>INKEY$7 ;6 UNPLOT RNDU▄RN
D/:5 GOSUB ?5 UNPLOT RNDU▙RND?U▌
RND▄E£RND7- ?▘S;( CLEAR U▙RND?;U
▛RNDM▙RND FAST U▜RND7)CINKEY$ GO
SUB ▩ LPRINT FAST )5  GOSUB ?SGN
 U▜RND? GOSUB ▩? GOSUB PIU▙RNDXM
▙RND4 SCROLL / INKEY$Q 7( CLS TR
N )CINKEY$ FOR GOSUB ▩TAN ......
.BBBBBBBBBBBBBBB................
...
  10 LET W$=CHR$ 3+CHR$ 3+CHR$ 1
5+CHR$ 6+"RA"
  20 FOR K=1 TO 10
  30 PRINT "AAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAA"
  40 PRINT "BBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBB"
  50 NEXT K
  60 RAND USR 16524
```

Listato 11. *Programma per l'implementazione dello scroll bidirezionale.*

```
16720 = 42          16740 = 1
16721 = 16          16741 = 4
16722 = 64          16742 = 0
16723 = 126         16743 = 17
16724 = 35          16744 = 130
16725 = 254         16745 = 64
16726 = 92          16746 = 237
16727 = 32          16747 = 176
16728 = 250         16748 = 229
16729 = 126         16749 = 205
16730 = 254         16750 = 148
16731 = 0           16751 = 64
16732 = 32          16752 = 225
16733 = 245         16753 = 193
16734 = 35          16754 = 43
16735 = 35          16755 = 43
16736 = 31          16756 = 16
16737 = 61          16757 = 237
16738 = 71          16758 = 201
16739 = 197
```

Dump 3. *Dump di controllo per inserire e verificare i codici del listato Assembly della figura 3.*

```
16518 = 0           16558 = 58
16519 = 42          16559 = 130
16520 = 16          16560 = 64
16521 = 64          16561 = 95
16522 = 125         16562 = 25
16523 = 35          16563 = 58
16524 = 254         16564 = 133
16525 = 92          16565 = 64
16526 = 32          16566 = 50
16527 = 250         16567 = 134
16528 = 126         16568 = 64
16529 = 254         16569 = 58
16530 = 4           16570 = 132
16531 = 32          16571 = 64
16532 = 245         16572 = 71
16533 = 35          16573 = 126
16534 = 35          16574 = [illegible]
16535 = 17          16575 = 128
16536 = 130         16576 = 119
16537 = 64          16577 = 35
16538 = 1           16578 = 16
16539 = 4           16579 = 249
16540 = 0           16580 = 58
16541 = 237         16581 = 132
16542 = 176         16582 = 64
16543 = 42          16583 = 95
16544 = 12          16584 = 62
16545 = 64          16585 = 33
16546 = 35          16586 = 147
16547 = 22          16587 = 95
16548 = 0           16588 = 25
16549 = 58          16589 = 58
16550 = 131         16590 = 134
16551 = 64          16591 = 64
16552 = 95          16592 = 61
16553 = 6           16593 = 200
16554 = 33          16594 = 50
16555 = 26          [illegible]
16556 = 16          16596 = 64
16557 = 253         16597 = 24
                    16598 = 226
```

Dump 4. *Dump contenente i codici della routine del listato 8 pubblicata disassemblata nella figura 4 e contenuta nelle REM del listato 8.*

PERSONAL SOFTWARE

Routine in linguaggio macchina per ZX81

```
16595 = 42
16596 = 132
16597 = 64
16598 = 22
16599 = 0
16600 = 69
16601 = 92
16602 = 33
16603 = 0
16604 = 0
16605 = 25
16606 = 16
16607 = 253
16608 = 6
16609 = 0
16610 = 235
16611 = 33
16612 = 0
16613 = 0
16614 = 120
16615 = 25
16616 = 16
16617 = 253
16618 = 235
16619 = 50
16620 = 16
16621 = 65
16622 = 42
16623 = 16
16624 = 64
16625 = 124
16626 = 254
16627 = 125
16628 = 40
16629 = 20
16630 = 126
16631 = 35
16632 = 254
16633 = 70
16634 = 32
16635 = 245
16636 = 126
16637 = 187
16638 = 32
16639 = 246
16640 = 35
16641 = 126
16642 = 186
16643 = 32
16644 = 241
16645 = 35
16646 = 34
16647 = 163
16648 = 64
16649 = 201
16650 = 62
16651 = 16
16652 = 50
16653 = 0
16654 = 64
16655 = 201
16656 = 0
16657 = 205
16658 = 135
16659 = 64
16660 = 34
16661 = 163
16662 = 64
16663 = 58
16664 = 16
16665 = 65
16666 = 61
16667 = 202
16668 = 211
16669 = 64
16670 = 50
16671 = 16
16672 = 65
16673 = 201
```

Dump 5. *Dump del programma per il trasferimento delle immagini su video, listato nella figura 5 e contenuto nella REM del listato 9.*

Dump 6. *Dump del programma per la protezione del software mediante chiave di accesso. Il listato Assembly è presentato nella figura 6. Il programma caricato appare nella REM del listato 10.*

```
16550 = 38
16551 = 39
16552 = 40
16553 = 41
16554 = 42
16555 = 229
16556 = 213
16557 = 205
16558 = 187
16559 = 2
16560 = 124
16561 = 198
16562 = 2
16563 = 56
16564 = 8
16565 = 68
16566 = 77
16567 = 205
16568 = 189
16569 = 7
16570 = 78
16571 = 58
16572 = 2
16573 = 14
16574 = 0
16575 = 209
16576 = 225
16577 = 201
16578 = 42
16579 = 12
16580 = 64
16581 = 35
16582 = 64
16583 = 146
16584 = 22
16585 = 0
16586 = 205
16587 = 171
16588 = 64
16589 = 121
16590 = 167
16591 = 40
16592 = 249
16593 = 61
16594 = 254
16595 = 117
16596 = 40
16597 = 23
16598 = 28
16599 = 122
16600 = 254
16601 = 20
16602 = 40
16603 = 11
16604 = 113
16605 = 35
16606 = 205
16607 = 171
16608 = 64
16609 = 121
16610 = 167
16611 = 32
16612 = 249
16613 = 24
16614 = 227
16615 = 205
16616 = 42
16617 = 10
16618 = 195
16619 = 194
16620 = 64
16621 = 122
16622 = 254
16623 = 5
16624 = 32
16625 = 245
16626 = 42
16627 = 12
16628 = 64
16629 = 17
16630 = 166
16631 = 64
```

Seguito dump 6.

```
16632 = 6
16633 = 5
16634 = 35
16635 = 78
16636 = 235
16637 = 126
16638 = 185
16639 = 32
16640 = 230
16641 = 35
16642 = 235
16643 = 16
16644 = 245
16645 = 205
16646 = 42
16647 = 10
16648 = 201
```

```
16518 = 62
16519 = 16
16520 = 50
16521 = 0
16522 = 64
16523 = 201
16524 = 42
16525 = 16
16526 = 64
16527 = 124
16528 = 254
16529 = 125
16530 = 40
16531 = 242
16532 = 126
16533 = 35
16534 = 254
16535 = 92
16536 = 32
16537 = 245
16538 = 126
16539 = 254
16540 = 6
16541 = 32
16542 = 245
16543 = 1
16544 = 4
16545 = 0
16546 = 35
16547 = 35
16548 = 17
16549 = 130
16550 = 64
16551 = 237
16552 = 176
16553 = 126
16554 = 254
16555 = 55
16556 = 40
16557 = 4
16558 = 62
16559 = 0
16560 = 24
16561 = 2
16562 = 62
16563 = 7
16564 = 50
16565 = 18
16566 = 65
16567 = 35
16568 = 126
16569 = 254
16570 = 39
16571 = 40
16572 = 11
16573 = 33
16574 = 0
16575 = 25
16576 = 34
16577 = 252
16578 = 64
16579 = 58
16580 = 131
16581 = 64
16582 = 24
16583 = 14
16584 = 33
16585 = 237
16586 = 62
16587 = 34
16588 = 252
16589 = 64
```

Dump 7. *Dump della routine in linguaggio macchina per lo scroll, listata nella figura 7 e contenuta nella REM del listato 11.*

Seguito dump 7.

```
16590 = 58
16591 = 131
16592 = 64
16593 = 95
16594 = 58
16595 = 133
16596 = 64
16597 = 131
16598 = 42
16599 = 12
16600 = 64
16601 = 35
16602 = 22
16603 = 0
16604 = 95
16605 = 6
16606 = 33
16607 = 25
16608 = 16
16609 = 253
16610 = 68
16611 = 130
16612 = 64
16613 = 95
16614 = 25
16615 = 58
16616 = 133
16617 = 64
16618 = 50
16619 = 130
16620 = 64
16621 = 229
16622 = 58
16623 = 132
16624 = 64
16625 = 79
16626 = 17
16627 = 40
16628 = 65
16629 = 237
16630 = 176
16631 = 225
16632 = 229
16633 = 17
16634 = 33
16635 = 0
16636 = 237
16637 = 82
16638 = 209
16639 = 58
16640 = 132
16641 = 64
16642 = 79
16643 = 237
16644 = 176
16645 = 79
16646 = 237
16647 = 66
16648 = 58
16649 = 130
16650 = 64
16651 = 61
16652 = 50
16653 = 130
16654 = 64
16655 = 32
16656 = 231
16657 = 24
16658 = 0
16659 = 65
16660 = 54
16661 = 0
16662 = 35
16663 = 16
16664 = 251
16665 = 201
16666 = 17
16667 = 40
16668 = 65
16669 = 235
16670 = 237
16671 = 176
16672 = 201
```

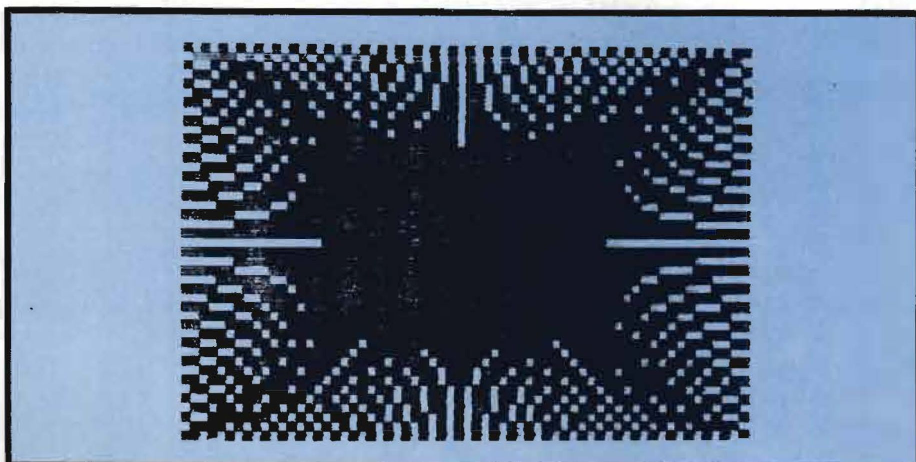


Immagine 1. *Immagine dell'unione dei punti periferici con il centro dello schermo, prodotta dal listato 2.*

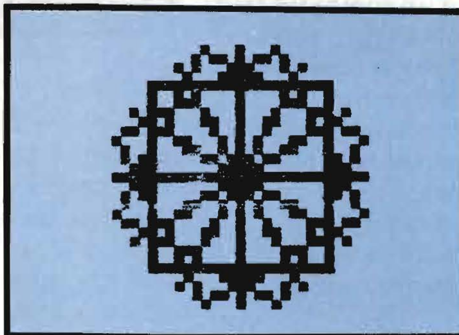
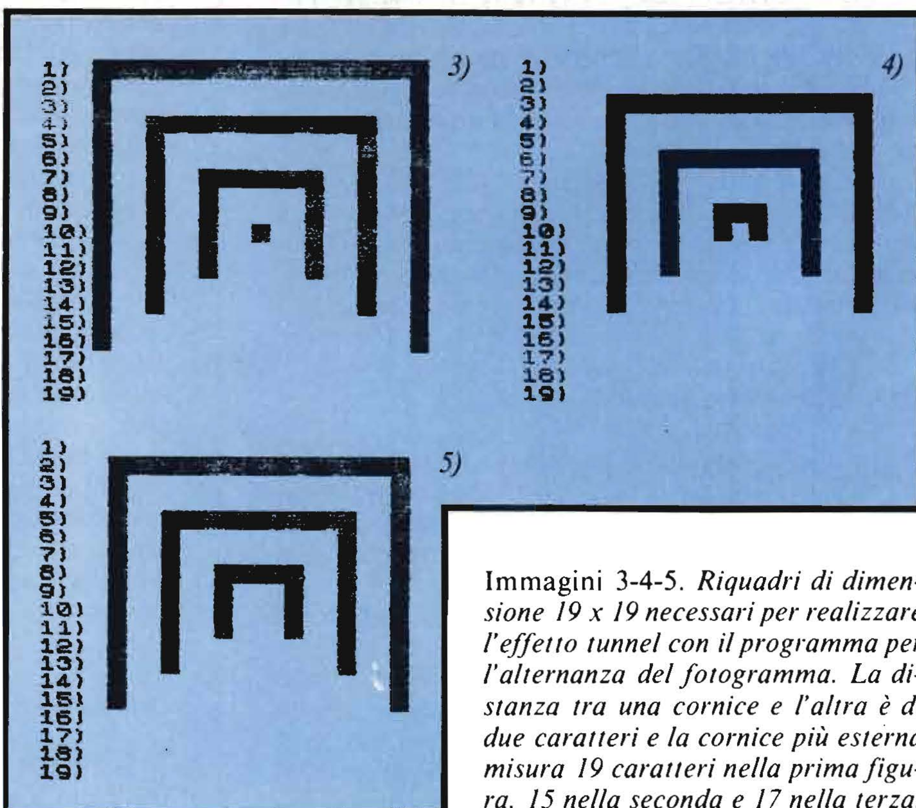


Immagine 2. *Esempio della rotazione di un quadrato con sfasamento degli argomenti trigonometrici.*



Immagini 3-4-5. *Riquadri di dimensione 19 x 19 necessari per realizzare l'effetto tunnel con il programma per l'alternanza del fotogramma. La distanza tra una cornice e l'altra è di due caratteri e la cornice più esterna misura 19 caratteri nella prima figura, 15 nella seconda e 17 nella terza.*

PERSONAL SOFTWARE

I SEGRETI DEI PERSONAL

TEXAS TI99/4A

Diagrammi in TI BASIC

di Sergio Borsani

Una delle più comuni rappresentazioni grafiche è data dai diagrammi a strisce o ortogrammi. Si tratta di rappresentare una serie di misure con delle strisce o delle colonne la cui lunghezza sia proporzionale alle misure stesse. Rispetto alla rappresentazione di risultati numerici sotto forma di tabella, un diagramma presenta il vantaggio di una maggiore immediatezza e consente di cogliere con un colpo d'occhio la proporzione tra i valori rappresentati o l'andamento di un fenomeno al quale essi si riferiscono. Il modo più semplice per risolvere il problema con un computer consiste nello stampare uno stesso simbolo ripetutamente in modo da formare qualcosa che assomigli ad una striscia. Alcuni computer, come il Commodore 64, posseggono già dei simboli grafici preprogrammati adatti allo scopo, con l'home computer è tuttavia semplicissimo programmare un carattere personalizzato e ripeterne la stampa in una qualsiasi posizione dello schermo. Se si desidera che le strisce risultino verticali si potrà definire il seguente carattere:

CALL CHAR (128, "7F7F7F7F7F7F7F7F")

Esso non occuperà l'intera matrice di 8 x 8 pixel ma possiede un margine a sinistra in modo che le strisce che formeranno il diagramma risultino leggermente separate una dall'altra. La stampa si ottiene poi con l'istruzione:

CALL VCHAR (R, C, 128, V)

dove R indica la riga e C la colonna, 128 indica il carattere precedentemente programmato e V il numero di volte in cui viene ripetuto lo stesso carattere, verticalmente, partendo dall'alto verso il basso.

Dimensionamento automatico

Un buon programma per la tracciatura di grafici dovrebbe possedere la caratteristica del dimensionamento automatico, cioè la striscia che rappresenta la misura maggiore dovrebbe automaticamente occupare la massima estensione consentita dal video, mentre tutte le altre risulteranno in proporzione più piccole.

Nell'esempio riportato nel listato 1 i valori variano secondo una legge esponenziale. La funzione definita alla linea 160 è la formula che si usa per calcolare l'interesse composto di un deposito in denaro. 1000 rappresenta il capitale iniziale, .14 è il tasso di interesse (14%) e T è il numero degli anni. Poiché il programma è fatto per stampare un grafico con 28 colonne è chiaro che in questo caso particolare il valore massimo da rappresentare si trova in corrispondenza di T = 28, cioè F (28).

```
100 REM  PROVA 1 ORTOGRAMMA
110 REM  ******************
120 CALL CLEAR
130 CALL CHAR(128,"7F7F7F7F7F7F7F7F")
140 CALL CHAR(136,"FF")
150 CALL COLOR(13,5,1)
160 DEF F(T)=1000*(1+.14)^T
170 MAX=F(28)
180 U=MAX/20
190 CALL HCHAR(22,3,136,28)
200 FOR J=1 TO 28
210 LUNG=INT(F(J)/U)
220 IF LUNG=0 THEN 240
230 CALL VCHAR(22-LUNG,J+2,128,LUNG)
240 NEXT J
250 PRINT TAB(7);"<BATTI UN TASTO>"
260 CALL KEY(0,K,S)
270 IF S=0 THEN 260
280 CALL CLEAR
290 END
```

Listato 1. *Programma "Prova 1 Ortogramma".*

È per questa ragione che alla linea 170 questo valore è stato preso come massimo. In altri casi si dovrà seguire una via diversa e andrà benissimo una piccola routine come quella riportata dal manuale a pagina 60. Chiamato MAX il valore massimo, se si desidera che la colonna più lunga non superi la lunghezza di 20 caratteri, sarà U = MAX/20 il valore unitario, cioè il valore di ogni tratto elementare che forma la striscia. La stampa dell'intero diagramma è compiuta dal ciclo FOR NEXT alle linee 200-240. LUNG è la lunghezza o il numero di caratteri di ogni striscia e l'istruzione 230 è fatta in modo che tutte le colonne abbiano la base posta in corrispondenza della riga 21 dello schermo. Eseguendo il programma del listato 1 si noterà che l'aspetto del diagramma non è molto soddisfacente. Il problema è che la lunghezza delle strisce varia per valori discreti abbastanza elevati e che misure differenti possono essere rappresentate con colonnine formate dallo stesso numero di caratteri. Il diagramma potrebbe essere migliorato se si potesse disporre di caratteri che rappresentano una parte frazionaria di quello fino ad ora usato.

Definiamo altri simboli grafici

Si osservi il listato 2. Oltre al carattere 128 ne troviamo altri sette (il carattere 136 serve solo a

```
100 REM  PROVA 2 ORTOGRAMMA
110 REM  ******************
120 CALL CLEAR
130 CALL CHAR(128,"7F7F7F7F7F7F7F7F")
140 CALL CHAR(129,"000000000000007F")
150 CALL CHAR(130,"0000000000007F7F")
160 CALL CHAR(131,"00000000007F7F7F")
170 CALL CHAR(132,"000000007F7F7F7F")
180 CALL CHAR(133,"0000007F7F7F7F7F")
190 CALL CHAR(134,"00007F7F7F7F7F7F")
200 CALL CHAR(135,"007F7F7F7F7F7F7F")
210 CALL CHAR(136,"FF")
220 CALL COLOR(13,5,1)
230 DEF F(T)=1000*(1+.14)^T
240 MAX=F(28)
250 U=MAX/20
260 CALL HCHAR(22,3,136,28)
270 FOR J=1 TO 28
280 LUNG=INT(F(J)/U)
290 IF LUNG=0 THEN 310
300 CALL VCHAR(22-LUNG,J+2,128,LUNG)
310 DIFF=F(J)/U-LUNG
320 PF=INT(8*DIFF)
330 IF PF=0 THEN 350
340 CALL HCHAR(21-LUNG,J+2,128+PF)
350 NEXT J
360 PRINT TAB(7);"<BATTI UN TASTO>"
370 CALL KEY(0,K,S)
380 IF S=0 THEN 370
390 CALL CLEAR
400 END
```

Listato 2. *Programma "Prova 2 Ortogramma".*

tracciare la base del diagramma). Ogni simbolo grafico occupa una matrice di 8 x 8 punti video (pixel), ebbene, il primo è alto 8 pixel e rappresenta il carattere unitario, gli altri sono alti rispettivamente 1, 2, 3, ..., 7 pixel. Ora che si posseggono i caratteri adatti, il nostro compito sarà quello di stampare per ogni striscia un certo numero di caratteri "unitari", come è stato fatto in precedenza, e, sopra questi, collocare il carattere "frazionario" più adatto. Senza riscrivere l'intero programma, è sufficiente aggiungere le linee dove vengono definiti i nuovi caratteri (vedi listato 2). Il programma procede poi allo stesso modo fino alla linea 300. Alla linea successiva si calcola la differenza (DIFF) tra il valore che si voleva rappresentare e quello che effettivamente si è rappresentato o meglio tra la lunghezza della colonna quale doveva essere e quale effettivamente è. Tale differenza è sempre inferiore all'unità ed è espressa in decimi. Tuttavia i simboli grafici definiti variano tra di loro di uno o più pixel ed un pixel rappresenta un ottavo di carattere, non un decimo. La linea 320 del programma trasforma la parte decimale in un numero compreso tra 0 e 7. Un esempio numerico chiarirà meglio questo punto. Se la differenza DIFF è uguale a 0.4, sarà PF = INT (8 ★ 0.4), cioè PF = 3 e alla linea 340 del programma si stampa il carattere 128 + PF, cioè il 131 che è alto i 3/8 del carattere unitario ed è quello che meglio rappresenta i 4 decimi mancanti alla colonnina. I valori utilizzati da questo programma sono gli stessi usati in precedenza ma la rappresentazione grafica è ora notevolmente più precisa.

COMMODORE 64

Il 64 e la memoria nascosta

di Alessandro Guida

È noto a tutti che il 64 è pubblicizzato dalla Commodore come un sistema da 64 Kbyte di RAM. Al contrario, all'accensione sullo schermo appare un deludente "38911 BYTES FREE".
Ma niente paura i restanti 24 Kbyte di RAM ci sono, anche se nascosti.

I banchi di memoria

Il microprocessore di cui è dotato il 64, il 6510, ha un bus indirizzi da 16 bit. Questo significa che è in grado di indirizzare 65536 locazioni, cioè 64 Kbyte di memoria. Di questi 64 Kbyte però, 16 sono riservati alla ROM contenente l'interprete BASIC e il sistema

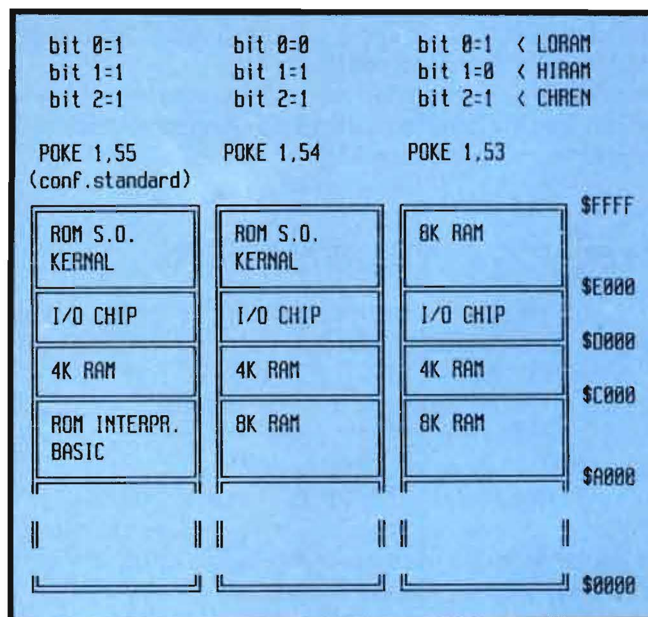


Figura 1. *Le tre più importanti configurazioni della memoria del C 64 con il relativo POKE nel registro 1.*

Il 64 e la memoria nascosta

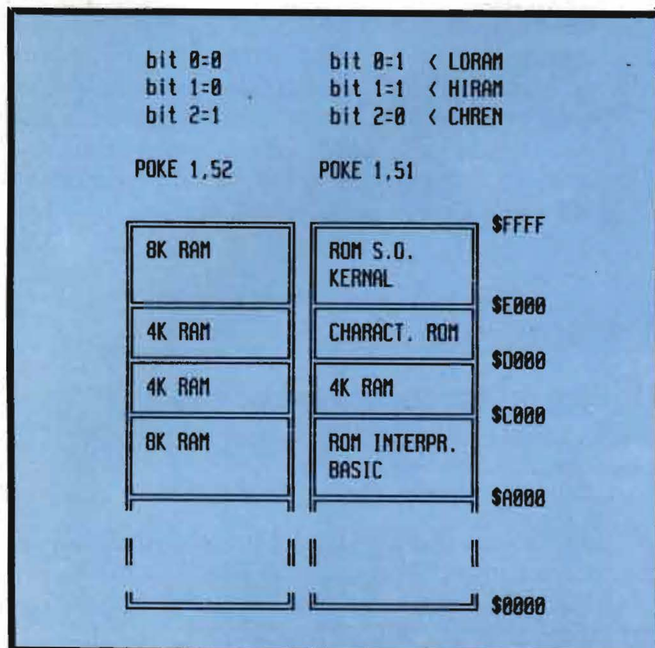


Figura 2. *Altre due possibili configurazioni della memoria del C 64 con il relativo POKE nel registro 1.*

operativo e altri 4 Kbyte sono a disposizione dei due circuiti CIA (Complex Interface Adapter) 6526. Perciò con i metodi tradizionali per la RAM rimarrebbero a disposizione solo 44 Kbyte.

Una soluzione, utilizzata su numerosi computer, per aumentare le capacità di memoria consiste nel mettere della memoria aggiuntiva in parallelo a quella già esistente. Gli indirizzi saranno quindi gli stessi, ma ci sarà una porta che si occuperà di selezionare un banco o l'altro di memoria.
Il microprocessore 6510 nasce già con tale porta, e si presta molto bene ad eseguire la gestione della memoria in modo "switching".

La memoria aggiuntiva del 64

Nel C 64 tale "memoria fantasma" è stata posta in corrispondenza delle ROM di sistema, a partire dall'indirizzo $A000 (40960) fino a $FFFF (65535). In effetti di questi 24 Kbyte di memoria quelli da $C000 (49152) a $CFFF (53247) sono sempre disponibili.
Possiamo verificare ciò facendo:

POKE50000, 127: PRINT PEEK (50000)

Otterremo come risposta 127 a significare che questa zona di RAM è perfettamente accessibile. Provate con altri valori per convincervene.

Poiché, però, questa zona è separata dalla memoria utilizzata normalmente dai programmi BASIC, diventa il luogo ideale per memorizzare routine in

```
C000                    ; ROUTINE LETTURA RAM > $A000
C000                    ;
C000                    ; $FB,$FC = indirizzo lettura
C000                    ; $FD = risposta
C000                    ; ---------------------------------
C000  78       SEI      ; Disabilita interrupt
C001  A5 01    LDA $01  ; Carica registro I/O micropr.
C003  48       PHA      ; Lo salva nello stack
C004  29 FD    AND &FD  ; Azzera bit 1
C006  85 01    STA $01  ; Modifica reg. I/O e abilita RAM
C008  A0 00    LDY &00  ;
C00A  B1 FB    LDA ($FB),Y ; Carica contenuto locazione
C00C                    ;  puntata da $FB,$FC
C00C  85 FD    STA $FD  ; Lo mette in $FD (253 dec.)
C00E  68       PLA      ; Recupera valore originale
C00F  85 01    STA $01  ;  registro I/O micropr.
C011  58       CLI      ; Riabilita Interrupt
C012  60       RTS      ; ---------------------------------
```

Figura 3. *Disassemblato della routine in linguaggio macchina presente nel programma del listato 1.*

linguaggio macchina che non verranno sicuramente rovinate dal BASIC stesso.

Tutto ciò non è valido per le altre zone occupate, normalmente, dalla ROM.
Se proviamo a dare:

POKE60000, 127: PRINT PEEK (60000)

avremo come risultato 235, a conferma di quanto detto.
È necessario perciò passare dalla ROM alla RAM corrispondente prima di potervi memorizzare dei dati.

La porta I/O del 6510

Abbiamo detto che la selezione dei banchi di memoria è effettuata dalla porta di I/O del 6510.
Questa si compone di due registri: un registro di direzione (DDR) e un registro dati. Fisicamente corrispondono, rispettivamente, alle locazioni 0 e 1 della memoria.
Il registro di direzione indica quali bit della porta sono di Input e quali di Output.
Se un bit di tale registro è uguale a 1, vuol dire che il corrispondente bit della porta sarà utilizzato per operazioni di output. Uno zero in un bit del DDR indica, invece, che il bit dello stesso ordine del registro dati sarà posto in ricezione.
Il valore normale del registro di direzione è:

00101111

pari è 47 in decimale. A parte il bit 4 gli altri predispongono la porta per operazioni di uscita dati.

In ogni caso, non dovrete preoccuparvi di questo registro, e se avrete il dubbio di averlo modificato accidentalmente sarà sufficiente dare POKE 0,47 per riportare tutto a posto.
Nel registro dati, i primi tre bit si occupano della selezione dei banchi di memoria, mentre gli altri sono



dedicati alla gestione del registratore.
Sarà quindi sufficiente fare un POKE nella locazione 1 per passare dalla ROM alla RAM corrispondente o viceversa.

Ognuno dei tre bit corrisponde ad un blocco di ROM/RAM. Il primo si riferisce a 8 Kbyte di RAM nascosta sotto l'interprete BASIC, da $A000 (40960) a $BFFF (49152); il secondo od altri 8 Kbyte di RAM sotto il sistema operativo, da $E000 (57344) a $FFFF (65535). Il terzo, infine, a 4 Kbyte di RAM in corrispondenza della ROM del generatore di caratteri.
Ecco il contenuto del registro 1:

bit	7 6 5 4 3 2 1 0
	0 0 1 1 0 1 1 1 valori standard (55 in dec.)
bit	significato
0	LORAM: 1 attiva ROM BASIC ($A000, $BFFF) 0 attiva RAM $A000, $BFFF
1	HIRAM: 1 attiva ROM S.O. KERNAL 0 attiva RAM $E000, $FFFF + LORAM
2	CHREN: 1 attiva CIA I/O 6526 ($D000, $DFFF) 0 attiva ROM generatore caratteri
3	Linea uscita dati per registratore
4	Interruttore tasto play registratore
5	Linea controllo motore registratore
6,7	Inutilizzati.

La selezione dei banchi

Come appena visto, all'accensione del 64 il registro 1 contiene il valore 55 in decimale che abilita le due ROM e i due CIA. Se si volesse selezionare la RAM da $A000, a $BFFF (LORAM) sarebbe sufficiente digitare:

POKE 1, 54

Invece, per attivare la HIRAM occorrerebbe dare:

POKE 1. 53

Nota: La selezione della RAM da $E000 a $FFFF comporta la selezione automatica, anche della LORAM.
La situazione per il blocco da 4 Kbyte, $D000-DFFF, è invece leggermente più complessa perché qui convivono tre tipi di memoria differenti.

Quando il bit 2 della porta I/O del 6510 è alto, è abilitata la lettura/scrittura nei registri dei CIA. Se, invece, il bit 2 è basso (POKE 1, 51) viene abilitata la ROM (ovviamente solo per lettura) contenente le immagini dei caratteri.
L'attivazione di tale ROM è utile quando si vogliono ricopiare alcuni caratteri in un'altra zona di RAM. Comunque il chip che si occupa della gestione dello schermo accede periodicamente a questa ROM, senza che l'utente se ne debba preoccupare, e indipendentemente dal contenuto del bit 2.

Per attivare, infine, la RAM compresa tra $D000 e DFFF è necessario rimettere a 1 il bit 2 e a 0 i bit 0 e 1.
In pratica si darà:

POKE 1, 52

In questa ultima configurazione si hanno 24 Kbyte di RAM continui da $A000 a $FFFF.
A questo punto, probabilmente, avrete già tentato di dare uno dei POKE appena visto in modo diretto. Ciò avrà provocato il blocco del sistema.

Niente paura, spegnete il 64 e riaccendetelo, tutto tornerà in ordine.
I POKE visti non è possibile darli in modo diretto, perché nel momento in cui viene disabilitato il BASIC o il sistema operativo, il computer nel tentare di eseguire le routine scritte nella ROM (ricezione caratteri, interpretazione comandi, gestione schermo) vi troverà, al loro posto, della RAM contenente dei dati assolutamente casuali. Risultato: blocco del sistema.

Prima di vedere come risolvere questo problema sottolineamo una caratteristica del 64. Se si esegue un POKE (o un STA in linguaggio macchina) in una zona contenente della ROM, il microprocessore conserverà i dati nella RAM corrispondente. Il problema di selezionare i vari banchi riguarda, quindi, la lettura e non la scrittura di dati nella RAM fantasma.
Riassumendo:
1) qualsiasi tentativo di lettura di una zona di memoria contenente della ROM, prima di averne selezionato la RAM, riporta il contenuto della ROM all'indirizzo specificato e non della RAM;
2) qualsiasi tentativo di scrittura, invece, modificherà la RAM sottostante e non la ROM.

Questa caratteristica intrinseca del sistema ci fornisce una possibile soluzione del problema accennato prima.

È possibile con un semplice ciclo leggere il contenuto dell'interprete BASIC o del sistema operativo e ricopiarlo nella RAM. In questa maniera, quando si seleziona la RAM il microprocessore non va in crisi poiché continua a trovare le routine che gli servono al loro posto. Il fatto che ora siano su RAM, invece che ROM, non fa alcuna differenza.
Se, ad esempio, vogliamo attivare la RAM da $A000 a $BFFF basterà la seguente linea a ricopiare tutto l'interprete BASIC:

```
FOR I = 40960 TO 49151: POKE I, PEEK (I):
NEXT I
```

L'istruzione POKE scrive nella RAM mentre il PEEK legge la ROM.
Adesso possiamo dare tranquillamente il comando:

POKE 1, 54

senza che il computer si blocchi. Se invece si vuole utilizzare la RAM da $E000 a $FFFF dovremo ricopiare anche il contenuto della ROM dall'indirizzo

I SEGRETI DEI PERSONAL

```
60000 REM ********************************
60001 REM *                              *
60002 REM * ROUTINE PER ACCEDERE ALLA    *
60003 REM * MEMORIA NASCOSTA DEL C64     *
60004 REM *                              *
60005 REM * VARIABILI UTILIZZATE         *
60006 REM *                              *
60007 REM *     AD =INDIRIZZO            *
60008 REM *     CO%=DATO LETTO           *
60009 REM *                              *
60010 REM * ALTRE FL%,PT                 *
60011 REM *                              *
60012 REM ********************************
60020 REM MEMORIZZA ROUTINE LA 1. VOLTA
60030 IFFL%=1THEN60050
60040 FL%=1:FORPT=0TO19:READA:POKE49152+PT,A:NEXT
60050 REM PASSA INDIRIZZO ALLA ROUT. LM
60055 IF(AD<40960OR(AD>49151ANDAD<57344))THENRETURN
60060 PT=INT(AD/256):POKE252,PT
60070 POKE251,AD-PT*256
60080 SYS(49152): REM LEGGE DATO DALLA RAM
60090 CO%=PEEK(253):RETURN
60095 REM DATA ROUTINE L.M.
60100 DATA120,165,001,072,041,250,133,001,160,000
60110 DATA177,251,133,253,104,133,001,088,096,000
```

Listato 1. *Routine per accedere alla RAM nascosta del 64.*

57344 fino al 65535.

Tutto ciò va bene se vogliamo, per esempio, apportare delle modifiche al BASIC. Non va, invece, assolutamente usato per memorizzare dei dati poiché, questi, modificherebbero le routine del BASIC o del S.O. provocando il blocco del computer.

```
10 FORJ=1TO20:A%=RND(1)*256
20 POKE41000+J,A%:PRINTA%;:NEXT
30 PRINT
40 FORJ=1TO20:AD=41000+J
50 GOSUB60000:PRINTCO%;:NEXT
60 END
```

Listato 2. *Programma esempio utilizzante la routine del listato 1.*

L'unica soluzione per utilizzare la RAM nascosta per conservarci dati è di scrivere una routine in linguaggio macchina.

Riportiamo, nel listato 1, un programma che può servire allo scopo, anche utilizzato come routine all'interno di altri programmi.

Il suo uso è molto semplice: basta mettere nella variabile AD l'indirizzo di cui si vuole leggere il contenuto e chiamare la routine. Il risultato sarà nella variabile CO%.

Per memorizzare i dati va semplicemente fatto un POKE nella locazione scelta.

Le zone di memoria utilizzabili sono: $A000, $BFFF (40960, 49151) e $E000, $F000 (57344, 65535). I 4 Kbyte da $D000 è preferibile non utilizzarli.

SHARP

Giochiamo con i numeri di linea

di Mauro Lenzi

Questa volta parleremo della gestione dei numeri di linea.

I numeri di linea occupano nel Text Buffer due byte; le locazioni corrispondenti al numero di linea della prima linea di un programma sono 45105 e 45106.

Digitiamo questo programma:

10 BEEP 3

Ora andiamo a vedere il contenuto delle due locazioni suddette: troveremo in 45105 il numero 224 ed in 45106 il numero 16.

Proviamo ora ad aumentare di uno il contenuto di 45105:

POKE 45105,225

Se andiamo a vedere il programma, troveremo, sorprendentemente:

110 BEEP 3

Aumentando ancora di uno il contenuto di questa locazione, troveremo che il numero di linea sarà diventato 210, e così via...

Possiamo quindi concludere che il valore delle centinaia di un numero di linea è definito dal primo byte secondo la formula:

centinaia = PEEK (primo byte del numero di linea) − 224

In esadecimale questi numeri risultano più mnemonici infatti 128 = &E0; così se vorremo una centinaia porremo nel primo byte del numero di linea &E1, se ne vorremo due porremo &E2, ecc..

Per esempio se digitiamo:

POKE 45105, &E9

la nostra istruzione BEEP 3 sarà finita alla linea 910.

Andiamo ora ad esaminare l'altro byte e digitiamo:

POKE 45106,17

il numero di linea sarà diventato 911.

Tuttavia le cose non sono così semplici, infatti se mettiamo in questo byte il numero 15, non ci troveremo, come potremmo aspettarci:

909 BEEP 3

ma una cosa molto più strana:

90 BEEP 3

Ancora una volta però i numeri esadecimali ci vengono in aiuto; immaginiamo di volere cambiare il numero di linea in 987: il byte delle centinaia è ancora a

posto, infatti la locazione 45105 contiene 133, che in esadecimale corrisponde a &E9, perciò dobbiamo cambiare solo la locazione 45106:

POKE 45106, &87

il nostro programma sarà diventato:

987 BEEP 3

Concludendo possiamo quindi dire che per cambiare le decine e le unità di un determinato numero di linea, dobbiamo modificare il secondo byte corrispondente a quel numero di linea secondo la formula:
decine ed unità = POKE secondo byte, & numero desiderato.

Dunque è sufficiente introdurre il numero desiderato, facendo finta che sia esadecimale, cioè senza trasformarlo realmente nel corrispondente numero esadecimale. Se vorremo che il nostro programma ridiventi:

10 BEEP 3

procederemo perciò così:
POKE 45105, &E0 (mette le centinaia a posto)
POKE 45106, &10 (mette a posto le decine e unità)

Qualche volenteroso potrà a questo punto, approfondendo un po' gli studi, realizzare un programma di Renumber. Io invece, ho in serbo ancora qualche sorpresa.
Proviamo a mettere in questi due byte cifre sballate:

POKE 45105, &EA

ed andiamo a vedere cosa è successo:

.10 BEEP 3

abbiamo ottenuto un numero di linea "decimale"!
Proviamo adesso:

POKE 45105, &EB

il risultato è:

E10 BEEP 3

che ne pensate di un numero di linea "esponenziale"? Naturalmente il più bello è che queste istruzioni funzionano senza problemi, infatti se diamo un RUN, possiamo udire chiaramente i tre Beep desiderati.

Lascio alla vostra fantasia e curiosità provare a immettere in queste due locazioni i numeri più curiosi e vi garantisco che troverete dei risultati molto strani e simpatici.
Io mi limito ad aggiungere questa "stranezza":
poniamo a zero il byte delle decine ed unità:

POKE 45106, 0

poniamo ora ad 1 il byte delle centinaia:

POKE 45105, 1

se ora andate a guardare il programma vi troverete:

01

senza che venga visualizzata l'istruzione BEEP 3, tuttavia dando un RUN potremo udire ugualmente i tre Beep: il programma c'è, ma non si vede!
Abbiamo così trovato un efficace sistema per proteggere i nostri programmi.

ZX SPECTRUM

LLIST e COPY in doppia altezza

di Marcello Spero

Questa puntata è dedicata alla folta schiera dei possessori di una ZX Printer, utilizzata in unione ad uno Spectrum. L'incredibile macchinetta, infatti, se da un lato lascia alquanto a desiderare sul piano della qualità di stampa, dall'altro offre possibilità illimitate di usi strani ed inconsueti, essendo totalmente dipendente, per il suo funzionamento, dal software esterno.
La routine che vi propongo ha due possibili usi, entrambi estremamente interessanti. Per prima cosa può produrre listati su stampante in doppia altezza, cioè con caratteri larghi otto pixel, come di consueto, ma alti sedici. Utilizzando la medesima routine, a

```
   1 REM

********************************
*   STAMPA IN DOPPIA ALTEZZA   *
********************************


versione per lo Spectrum 16 K

puo' essere rilocata modificando
le istruzioni CLEAR e LET
in linea 10

  10 CLEAR 32519: LET y=32520
  20 READ x: IF x<>-1 THEN POKE
y,x: LET y=y+1: GO TO 20
  30 FOR x=3759 TO 3763: POKE y,
PEEK x: LET y=y+1: NEXT x
  40 FOR x=3762 TO 3768: POKE y,
PEEK x: LET y=y+1: NEXT x
  50 FOR x=3764 TO 3785: POKE y,
PEEK x: LET y=y+1: NEXT x
  60 READ x: IF x<>-1 THEN POKE
y,x: LET y=y+1: GO TO 60
  70 STOP
  80 DATA 62,1,24,28,42,83,92,22
9,205,175,13,225,126,230,192,192
,205,184,25,213,205,85,24,33,137
,92,62,25,150,135,135,135,71,243
,-1
  90 DATA 224,62,4,211,251,251,2
4,188,-1
 100 REM

            ************
            *ISTRUZIONI*
            ************

 110 REM

          POKE 32521,4:
          RANDOMIZE USR 32520

          stampa le prime 4 linee
          di pixel dello schermo.

 120 REM

          RANDOMIZE USR 32524

          lista l'intero programma
```

Listato 1. *Il programma di caricamento.*

PERSONAL SOFTWARE

LLIST e COPY in doppia altezza

```
10 CLEAR 32519: LET y=32520
20 READ x: IF x<>-1 THEN POKE
y,x: LET y=y+1: GO TO 20
30 FOR x=3759 TO 3763: POKE y,
PEEK x: LET y=y+1: NEXT x
40 FOR x=3762 TO 3768: POKE y,
PEEK x: LET y=y+1: NEXT x
50 FOR x=3764 TO 3785: POKE y,
PEEK x: LET y=y+1: NEXT x
60>READ x: IF x<>-1 THEN POKE
y,x: LET y=y+1: GO TO 60
70 STOP
80 DATA 62,1,24,28,42,83,92,22
9,205,175,13,225,126,230,192,192
,205,184,25,213,205,85,24,33,137
,92,62,25,150,135,135,135,71,243
,-1
90 DATA 224,62,4,211,251,251,2
4,188,-1
```

Listato 2. *Parte del programma di caricamento riprodotto in doppia altezza.*

partire da un diverso punto di ingresso, possiamo inoltre ottenere copie su carta (hard copy) dell'intero schermo o di parte di esso, sempre in doppia altezza: una COPY ingrandita nel senso dell'altezza e, volendo, parziale.

Veniamo ora ad un esame più approfondito di ciascuna delle due funzioni. Per quanto riguarda la prima, il suo uso è facilmente intuibile; la stampantina Sinclair, infatti, riesce spesso a produrre caratteri che sanno più di "geroglifici" che di qualcosa di comprensibile. In questi casi, specialmente se ad essere poco chiare sono le linee contenenti dati, o addirittura linee DATA, possono sorgere dei problemi. Produrre stampe ingrandite nel corso di un programma non è un problema usando il BASIC ed i caratteri grafici definibili; diversa è invece la situazione nel caso dei listati. Le linee che compongono un programma, infatti, contengono oltre ai caratteri che vediamo riprodotti sullo schermo quando ne otteniamo il listato, numerosi caratteri non stampabili, con funzione di controllo o per l'indicazione del valore delle costanti numeriche in virgola mobile. Per questo è consigliabile l'uso del linguaggio macchina.

Il programma visualizza una per volta le linee del programma in cima allo schermo, e quindi le riproduce usando una versione modificata della routine di COPY presente in ROM.

Questo procedimento, apparentemente macchinoso, è necessario per eliminare i caratteri non stampabili di cui dicevamo prima. Per la visualizzazione sullo schermo, infatti, viene usata la routine ROM adoperata normalmente dal sistema per questo scopo, che esegue automaticamente l'eliminazione. A questo punto interviene la routine di COPY, che differisce dall'originale per il fatto che ciascuna riga di pixel (ogni linea ne contiene otto) viene copiata due volte, realizzando così la doppia altezza.

Dopo che ciascuna linea è stata riprodotta, lo schermo viene ripulito e il processo si ripete con la linea successiva, fino all'ultima. Da questa descrizione del procedimento usato emerge l'unica limitazione del sistema, e cioè l'impossibilità di riprodurre correttamente linee di programma che siano più lunghe di un intero schermo; mi sembra comunque che si tratti di una limitazione più teorica che pratica, poiché, sebbene sia tecnicamente possibile realizzare linee di programma di qualsiasi lunghezza, tutti coloro che hanno tentato di manovrare linee di lunghezza superiore a dieci righe sanno come qualsiasi operazione, sia essa di aggiunta di caratteri o di correzione, diventi talmente lenta da risultare esasperante. Manipolare una linea che si estende oltre i limiti dello schermo, oltretutto, significa non vedere ciò che stiamo scrivendo, come ci viene fatto notare dal sistema con l'apposito segnale acustico (il manuale lo chiama RASP, che vuol dire gracidìo ...).

Per ottenere la funzione di LLIST ad altezza doppia di un programma basta dare RANDOMIZE 32524 se (ed è questo il caso dello Spectrum 16 Kbyte) avete caricato la routine all'indirizzo indicato nel programma di caricamento, o "primo byte della routine" più quattro se avete collocato altrove il programma (che è infatti completamente rilocabile, come vedremo più avanti). Durante il funzionamento è possibile interrompere in qualsiasi momento, mediante il BREAK, la stampa; ovviamente sarà poi necessario ricominciare tutto da capo.

Come abbiamo detto all'inizio, la medesima routine può essere utilizzata per copiare totalmente o parzialmente quanto è presente sul video, ottenendone una copia ad altezza doppia. Per far questo occorre accedere alla routine da un altro punto di inizio, dopo aver determinato mediante una POKE quante righe di pixel vogliamo riprodurre. La nostra scelta può spaziare da una sola riga, nel qual caso otterremo due righe stampate, fino a 176, cioè l'altezza dell'intero schermo. Nel caso vi dimentichiate di specificare questo valore verrà preso per buono l'ultimo usato, o il valore uno nel caso la routine sia stata appena caricata.

Per ottenere questa funzione è necessario, naturalmente dopo aver riempito lo schermo con quanto

LLIST e COPY in doppia altezza

```
ENTR_1  ld a,01        ;ingresso
                       ;COPY
        jr,MAIN        ;salta al
                       ;prog. princ.
ENTR_2  ld hl,(PROG)   ;ingresso
                       ;LLIST. hl
        push hl        ;punta all'
                       ;inizio prog.
LOOP    call CL-ALL    ;<0DAF>
                       ;esegue CLS
        pop hl         ;punta alla
                       ;linea corr.
        ld a,(hl)      ;il programma
                       ;e' finito?
        and C0         ;se si' ritor
                       ;na al Basic.
        ret nz         ;
        call NXT-ONE;  <1988>
                       ;altrimenti
        push de        ;de punta
                       ;alla linea
        call OUT-LIN;  <1855>
                       ;PRINT linea
        ld hl,(SPOS)   ;punta all'
                       ;ultima linea
                       ;occupata,
        ld a,19        ;complementa
                       ;a 25 e molti
        sub(hl)        ;plica per 8
                       ;ottenendo il
        add a,a        ;n. di pixel
        add a,a
        add a,a
MAIN    ld b,a         ;b e' il cont
                       ;dei cicli
        di             ;disabilita
                       ;le interruz.
        ld hl,4000     ;l'inizio del
                       ;DISPLAY FILE
COPY-1  push hl        ;salva due
                       ;volte il pun
        push bc        ;tatore alla
                       ;linea e il
        push hl        ;contatore di
                       ;linee di
        push bc        ;pixel
        call CPY-LIN;  <0EF4>
                       ;copia su
        pop bc         ;stampante
                       ;una riga di
        pop hl         ;pixel per
                       ;due volte,
        call CPY-LIN;  recuperando
                       ;ogni volta
        pop bc         ;hl e bc
        pop hl
        inc h          ;modifica hl
                       ;perche'
        ld a,h         ;punti alla
                       ;riga success.
        and 07         ;se h<7 h=h+1
        jr nz,COPY-2;  salta avanti
        ld a,l         ;altrimenti
                       ;aggiorna hl
        add a,20       ;a seconda
                       ;del "terzo"
        ld l,a         ;di schermo
                       ;occupato
        ccf
        sbc a,a
        and F8
        add a,h
        ld h,a
COPY-2  djnz COPY-1    ;salta se le
                       ;righe non
        ld a,04        ;sono ancora
                       ;finite
        out(FB),a      ;se no ferma
                       ;il motore
        ei             ;abilita le
                       ;interruzioni
        jr LOOP        ;salta indie-
                       ;tro a LOOP

nota: i numeri fra <> sono gli
      indirizzi delle corrispon-
      denti routine di sistema
```

Listato 3. *Assembly della routine.*

vogliamo riprodurre, specificare innanzitutto il numero di righe da riprodurre, con POKE 32521, n sempre se la collocazione della routine è quella indicata, o POKE "inizio routine" più uno per qualsiasi altra collocazione.

Con n indichiamo il numero di pixel, numero che, come abbiamo visto, può variare da uno a 176. Anche in questo caso è attivo il BREAK, che può quindi essere usato in qualsiasi momento.

Come potete osservare dal confronto fra le figure 1 e 2, questa tecnica di riproduzione offre interessanti possibilità grafiche; il listato 2, invece, evidenzia l'aumento di chiarezza che si ottiene con un listato in doppia altezza. Il confronto è facile, trattandosi del programma di caricamento, lo stesso riprodotto nel listato 1, oltretutto contenente linee DATA.

Vediamo adesso il programma di caricamento. La sua prima linea contiene l'istruzione CLEAR per l'abbassamento della RAMTOP e l'istruzione LET che definisce la locazione di inizio della routine. I valori dati si riferiscono alla miglior posizione possi-

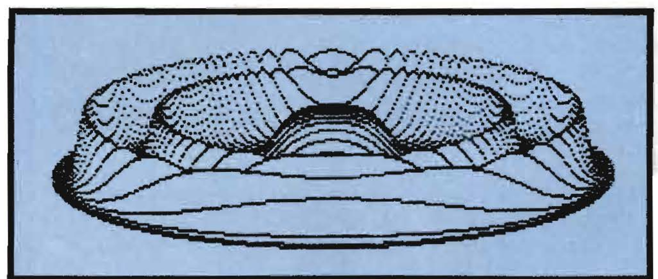


Figura 1. *Hard copy normale di un grafico.*

bile per la versione da 16 Kbyte, e cioè subito sotto l'area della grafica definibile. La routine, come accennato prima, è comunque completamente rilocabile, poiché non fa uso di istruzioni di salto assolute; basta quindi variare i valori della CLEAR e della LET per collocarla nella posizione più opportuna, in particolare nel caso possediate la versione da 48 Kbyte.

Il programma appare, nel suo insieme, alquanto inconsueto. Ciò è dovuto al fatto che, come abbiamo



LLIST e COPY in doppia altezza

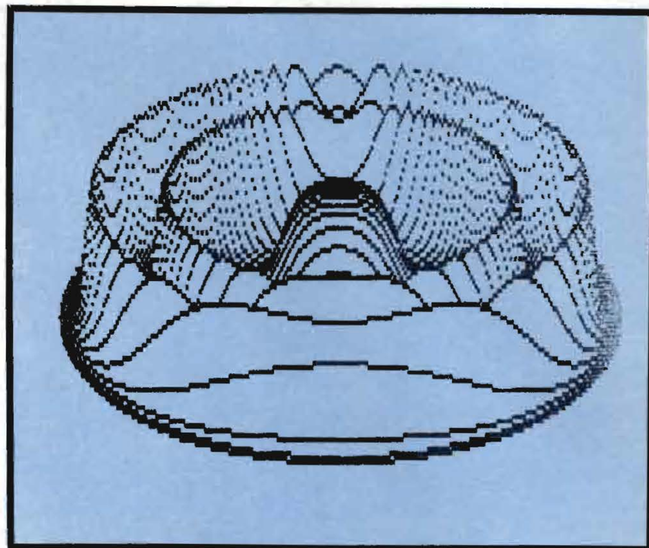


Figura 2. *Hard copy in doppia altezza dello stesso grafico.*

detto, la routine di COPY differisce dall'originale contenuta nella ROM solo in alcuni punti. Per rendere più semplice il caricamento si è perciò pensato di copiare semplicemente dalla ROM le parti uguali, caricando dalle DATA solo le parti modificate.

Dopo aver ricopiato (e controllato ...) il programma vi consiglio di salvarlo subito su cassetta, prima di provarne il funzionamento, per due ragioni. Se qualcosa, infatti, dovesse andare storto, per un errore di trascrizione, molto probabilmente la macchina andrà in "crash" e sarete costretti a spegnerla, perdendone il contenuto; salvandolo su cassetta non sarete costretti a ricopiare nuovamente tutto il programma, ma potrete limitarvi a cercarne gli errori. La seconda ragione riguarda invece la necessità che la routine venga conservata nella sua forma originale, cioè senza le modifiche introdotte con le POKE; in questo modo il valore corrispondente al numero di righe di pixel da copiare sarà, in assenza di una nostra specifica, uguale ad uno (qualcuno ama dire che il valore di "default" è uno).

A questo punto non vi resta che dare RUN per attuare il caricamento. Provatene quindi il funzionamento, e se tutto è a posto date NEW per eliminare il programma di caricamento, ormai inutile; basterà infatti dare le apposite RANDOMIZE per ottenere le operazioni volute.

Per coloro che fossero particolarmente interessati all'architettura del linguaggio macchina usato, il listato 3, riporta l'assembly con i relativi commenti, comprendenti gli indirizzi, all'interno della ROM, delle routine di sistema richiamate. I rispettivi punti di ingresso sono ENTR 1 per la COPY, e ENTR 2 per la LLIST.

PERSONAL SOFTWARE

CONTRIBUTI DEI LETTORI

Mini monitor per ZX81

Il lettore Marino Cappelletti di Como ha inviato questo breve ma utile programma che permette di visualizzare agevolmente il contenuto della memoria in codice esadecimale.
Il "monitor" vero e proprio è costituito dalla routine in linguaggio macchina del listato 1, che deve essere caricata a partire dall'indirizzo 16514 e occupa 87 byte. Essa produce la stampa sul video del contenuto di 66 locazioni di memoria, come in figura 1, a partire da quella indicata dal contenuto delle locazioni 16445, 16446.
Il programma del listato 2 serve da caricatore e, dopo avere inserito i codici del listato 1, occorre cancellare le linee da 20 a 60 (senza dare il NEW!) e inserire il programma del listato 3. Dopo il RUN il programma chiede l'indirizzo della prima locazione di memoria da cui iniziare il "dump" e poi accetta i comandi "6" o "7" per lo scrolling locazione per locazione, "5" o "8" per esaminare il precedente od il successivo blocco di 66 locazioni, "C" per copiare il contenuto del video su stampante e "N" per inserire un nuovo indirizzo di partenza.

```
2A 0C 40 11 09 00 19 EB 2A
3D 40 01 03 16 E5 C5 01 02
03 7E 32 3C 40 21 3C 40 3E
00 ED 67 C6 1C 12 1B 0D 20
F5 0E 02 3E 03 B8 20 08 3E
16 12 1B 12 1B 12 1B 23 10
E3 C1 05 28 0C 21 2A 00 19
EB E1 23 22 3D 40 18 C7 06
16 0D 28 09 EB 11 A1 02 A7
ED 52 18 EA E1 C9
```

Listato 1. *Codici esadecimali del "Mini monitor". Occorre usare questi codici come input per il programma del listato 2.*

```
  10 REM 0000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
  20 FOR N=16514 TO 16600
  30 INPUT H$
  40 POKE N,16*CODE H$+CODE H$(2
)-476
  50 NEXT N
  60 PRINT "O.K."
```

Listato 2. *Caricatore per i codici esadecimali del listato 1. È possibile usare questo programmino anche per caricare i vostri codici macchina.*

Listato 3.

```
  20 SLOW
  30 INPUT H$
  40 IF LEN H$<>4 THEN GOTO 30
  50 LET D=CODE H$*4096+CODE H$(
2)*256+CODE H$(3)*16+CODE H$(4)-
122332
  60 POKE 16446,INT (D/256)
  70 POKE 16445,D-256*INT (D/256
)
  80 RAND USR 16514
  90 IF INKEY$="5" THEN LET D=D-
66
 100 IF INKEY$="6" THEN LET D=D-
1
 110 IF INKEY$="7" THEN LET D=D+
1
 120 IF INKEY$="8" THEN LET D=D+
66
 130 IF INKEY$="C" THEN COPY
 140 IF INKEY$="N" THEN GOTO 30
 150 GOTO 60
```

Listato 3. *Master che gestisce l'input dell'indirizzo in notazione esadecimale e le funzioni di scrolling a locazioni singole oppure a blocchi.*

```
4000---FF   4016---BD   402C---FF
4001---C0   4017---41   402D---02
4002---FC   4018---06   402E---69
4003---7C   4019---C0   402F---D2
4004---00   401A---77   4030---87
4005---7D   401B---45   4031---0C
4006---00   401C---77   4032---00
4007---50   401D---45   4033---16
4008---00   401E---40   4034---13
4009---00   401F---5D   4035---ED
400A---14   4020---40   4036---00
400B---00   4021---00   4037---00
400C---50   4022---02   4038---BC
400D---42   4023---00   4039---21
400E---51   4024---00   403A---18
400F---42   4025---FE   403B---C0
4010---69   4026---EF   403C---00
4011---45   4027---0F   403D---3D
4012---08   4028---37   403E---40
4013---41   4029---BE   403F---00
4014---77   402A---41   4040---00
4015---45   402B---FE   4041---00
```

Figura 1. *Esempio di stampa prodotta dalla routine linguaggio macchina del listato 1.*

Conversione veloce decimale-binario

Quante volte si cerca una routine di conversione da inserire nei programmi più disparati! Di solito se ne inventa una sul momento, ma si rischia di cadere nell'inefficienza.

Marco Spada di Milano ci ha suggerito questa veloce routine di conversione da decimale a binario che funziona con qualsiasi numero di bit. Il punto di maggiore interesse è l'uso delle linee 14, 50, 65 e 70 invece del solito ciclo FOR ... NEXT, e la conseguente uscita dal loop quando il risultato parziale è 0, che velocizza la conversione. Si noti anche che il dimensionamento del vettore alla linea 16 viene ripetuto per ogni numero al fine di azzerare il contenuto del vettore stesso. Il programma gira su Spectrum.

```
  10 INPUT "numero da convertire
";n
  12 IF n>65535 OR n<0 THEN PRIN
T "numero non accettato": GO TO
10
  13 REM conversione
  14 LET n=INT (n+.5)
  16 DIM b(16)
  18 LET x=1
  20 LET p=INT (n/2)
  30 LET c=p*2
  40 LET b(x)=n-c
  50 LET x=x+1
  60 LET n=p
  65 IF n=0 THEN GO TO 80
  70 GO TO 20
  75 REM stampa
  80 FOR i=16 TO 1 STEP -1
  90 PRINT b(i);
 100 NEXT i
 110 PRINT
 120 GO TO 10
```

Listato 1. *Routine di conversione decimale-binario.*

SERVIZIO SOFTWARE



PERSONAL SOFTWARE

P.S. propone ai propri lettori i dischi o le cassette dei programmi pubblicati. I programmi, provati e garantiti, sono di immediato utilizzo.

P.S. n°	Programma	Sistema	Prezzo	Codice	Supporto
3	La carta del cielo Collisione	Apple II	30.000	1	Disco
2	Editor/Assembler in BASIC	CBM 3032	40.000	3	Disco
4	Interi in precisione multipla Grafica 3D	Apple II	40.000	4	Disco
4	Gioco del calcio	CBM 3032	25.000	5	Disco
5	Pretty printer Shape table	Apple II	30.000	6	Disco
7	Data base modulare	Apple II	25.000	7	Disco
12-13	Wei-ch'i	CBM 3032	20.000	8	Cassetta
14	Tool-Kit	C 64	35.000	9	Cassetta

Per richiedere i programmi in contrassegno, pagando direttamente al postino la cifra indicata, inviare il seguente tagliando

Spedire in busta chiusa a Gruppo Editoriale Jackson - Via Rosellini, 12 - 20124 Milano

Inviatemi i seguenti nastri e/o dischi con i programmi pubblicati su P.S.



GRUPPO EDITORIALE JACKSON

Cod. ☐☐☐☐☐☐ **a L.** ....

Cod. ☐☐☐☐☐☐ **a L.** ....

Cod. ☐☐☐☐☐☐ **a L.** ....

Cod. ☐☐☐☐☐☐ **a L.** ....

Spese postali (contributo fisso) L. 2.000

TOTALE L. ....

che pagherò al postino alla consegna del pacco.

Cognome ....

Nome ....

Indirizzo ....

CAP ....

Città ....

Firma ....

PICCOLI ANNUNCI

Apple

Cambio/vendo programmi per Apple II. Dispongo di vasto software. Rispondo a chiunque mi contatta. Telefonare ore serali 0481/81254 o scrivere a Vittor Franco.
Franco Vittor - Via Grabizio, 35 - 34170 Gorizia - Tel. 0481/81254.

Vendo software per Apple II di tutti i tipi o scambio con hardware. Scrivere per lista a:
Marco Carrubba - Via M. Campionesi, 29 - 20135 Milano - Tel. 02/585294.

Scambio moltissimi programmi di giochi per Apple, circa un centinaio. L'offerta vale solo per Roma.
Karim Sako - Via Lazzaro Spallanzani, 4/6 - 00161 Roma - Tel. 06/3275666.

Cambio/vendo programmi per Apple di tutti i tipi. Inviate la vostra lista, risponderò con la mia.
Carlo Casalicchio - Via M. Velino, 24 (Residence Ameno) - 60100 Ancona

Vendo Apple II compatibile, monitor 9", language 16 kb, drive, controller, interf. parallela, stampante Oki 82 grafica. Il sistema L. 3.500.000. Vendo Sharp PC 1211 con stampante CE122 L. 270.000.
Fortunato Petrocelli - C.so Lione, 44 - 10141 Torino.

Vendo possibilmente in blocco programmi Apple II gestionali, ingegneria grafica, data base, utilities, tantissimi giochi. Dispongo anche di parecchi manuali. Richiedere elenco o telefonare a:
Maurizo De Gioanni - C.so S. Santarosa, 67 - 12100 Cuneo - Tel. 0171/61839.

Vendo/cambio programmi per Apple (giochi, utilità, gestionali, ecc.), anche su commissione. Scrivere per informazioni (Residence Colle Meno).
Carlo Casalicchio - Via Velino, 24 - 60100 Ancona.

Commodore

Vendo modulo per inserire tre cartucce e/o espansioni nel VIC 20 L. 70.000. Penna ottica L. 50.000. Scambio programmi per VIC 20.
Riccardo Cappa - Via Tetti None, 1 - 10090 Baldissero T.se (TO) - Tel. 9408954 (ore pasti).

Scambio/vendo/compro programmi per C64.
Antonio Juliano - Via Rossi, 74 - 80056 Ercolano (NA) - Tel. 081/7395987 (ore serali).

Vendo PET/CBM 2001 Commodore causa cambio sistema completo di monitor, registratore 8 kRAM, 12 kROM, 4 libri sul PET, 8 cassette di software + sound box imballaggio originale; tutto per L. 500.000 trattabili.
Angelo Vianello - Via S. Croce, 2158 - 30100 Venezia - Tel. 041/25723.

Vendo/scambio software VIC 20 su cassetta. Posseggo numerosi programmi in L.M. Posseggo anche listati di programma d'ogni tipo per ZX 81, Spectrum, Texas, Apple. Scrivere e/o telefonare ore serali a:
Christian Ferlini - Via Aeutine, 32 - 00121 Ostia Lido - Tel. 06/5697207 (ore serali)

Vendo/compro/cambio programmi per VIC 20 in configurazione base sia su cassetta che listati. Prezzi modici. Rispondo a tutti. Per scambio programmi inviatemi vostra lista, io invio la mia a tutti.
Maurizio Mercati - Via Risorgimento, 8 - 57025 - Piombino (LI) - Tel. (0565) 35875.

Vendo a L. 100.000 alimentatore stabilizzato 0 ÷ 30 volts con due strumenti oppure cambio con espansione di memoria per VIC 20.
Daniele Vasi - Via Ofanto, 9 - 48100 Ravenna - Tel. 0544/63453.

Compro per C64 programmi di ogni genere solo se su cassetta e con particolare riguardo al software riguardante musica. Inviare liste.
Giuseppe Moliterni - Via Vico Fornaci, 8 - 75100 Matera - Tel. 0835/211874 (ore pomeridiane).

Vendo VIC 20 + registratore 1530 C2N + alimentatore + modulatore RF + cassetta mole attack + libro guida; tutto a L. 400.000, usato poco, due mesi di vita. Possibilmente zona Roma.
Giovanni Nitti - Via S. Giuliano-Terme, 145 - 00148 Roma - Tel. 06/5234908 (ore 20-21).

Vendo meravigliose penne ottiche con soft. Commodore 64, inoltre dispongo centinaia di programmi, di giochi e di utilità tra cui Pascal, Logo, Pilot, Forth, Macroas, Simon, Bas 40, Exbas2.2, Lastone, ecc.
Benedetto De Benedetti - V.le Curreno, 57/1 - 10133 Torino - Tel. 011/6509128.

Compro per VIC 20 configurazione base, listati di qualsiasi tipo di programma meglio in L.M. per giochi, a bassissimo costo. Inviatemi le vostre liste con relativo prezzo.
Andrea Mamini - Via Cantagalli, 19 - 48018 Faenza (Ravenna) - Tel. 25474.

Vendo/cambio programmi su disco o cassetta per Commodore 64. Scrivere o telefonare per lista ore ufficio. Giochi originali americani e utilities.
Vincenzo Coscia - V.le G. Marconi, 40/A - 80059 Torre Del Greco - Tel. 081/7011988.

Compro/cambio/vendo programmi per Commodore 64. Cerco inoltre programma "nukewar". Scrivere per elenco programmi o per contatti.
Sergio Fattorini - Via Monte alla Rena, 17 - 57016 Rosignano Solvay (LI).

Per sfruttare il vostro CMB64 vi occorre software potente con testi specializzati. Cedo alla metà del prezzo di listino: Simon's, Basic, Easy, Script 64, Diary, Word, Protrhee, Plus.
Luca Montalbano - Via Malagrida, 14 - 65100 Pescara - Tel. 085/34196.

Compro software scientifico per il Commodore 64. Cerco manuali possibilmente italiani sul 64.
Mauro Fallanca - Via Dante, 28 - 18039 Ventimiglia - Tel. 0184/355590.

Cerco CBM 64 purchè 6581 integro e periferiche sempre per CBM 64. Cerco anche solo 6581 (integrato musica).
Aldo Iocco - Via Ciccotti, 10 - 85100 Potenza - Tel. 0971/20468 (ore pasti).

Vendo Scacchi, Spriteman, Frogger, Scramble 64, Panic 64, Startrek, Calcio, Biliardo, Skirace. Per CBM 64 Monopoly a L. 15.000 cad. inoltre vendo Simons Basic (su nastro) a L. 45.000 con fotocopie istruzioni.
Antonio Munari - Via Dei Mille, 4 - 35100 Padova - Tel. 049/660227.

Vendo/cambio per VIC 20 programmi ad alta risoluzione grafica, come Pac-man, Bonzo, Scramble e altri 200 circa. Posseggo anche di utility, matematici, sonori e dimostrativi, il tutto a prezzi modesti.
Antonio Liuni - Via Parco dei Principi, 50 - 70010 Casamassima (Bari) - Tel. 080/671708 (dopo ore 20).

Compro software per VIC 20 (max 5 k) faccio affari solo per telefono e nella zona di Roma.
Diego Bardari - Roma - Tel. 06/8123684 (ore 14-15/20-21).

Vendo/scambio/compro programmi per Commodore 64. Posseggo inoltre 300 programmi, in particolare videogiochi, rispondo a tutti.
Daniele Forcella - Via Carducci, 83 - 65100 Pescara.

Vendo per VIC 20: 3 k RAM Super Expander L. 60.000; Cartuccia "The sky is falling" L. 20.000; libro "Guida al VIC 20" della E.V.M. L. 20.000; modulo di espansione VIC 1020 al miglior offerente.
Walter Collini - Via Frassineto - 52040 Arezzo - Tel. 0575/367105 (dopo ore 15).

Cambio/vendo programmi di giochi per Commodore 64. Solo zona La Spezia.
Alessandro Mecchia - Via V. Veneto, 168 - 19100 La Spezia - Tel. 0187/37247.

Vendo programmi per Commodore 64.
Paolo Bigon - Via San Marco, 177/A - 35020 Padova - Tel. 049/629117.

Vendo VIC 20 + espansione grafica e RAM "Super expander" + il libro "Alla scoperta del VIC 20" + interfaccia per qualunque registratore. Il tutto a sole L. 240.000 (prezzo del VIC inespanso).
Alberto Ferrari - Via S. Michele Del Carso, 22 - 10135 Torino - Tel. 011/341119.

Vendo per C-64 calcio della Commodore in 3-D e A.M.C. (Attack of Mutant Camels). Tutti e due interamente in L.M. su cassetta a L. 15.000. Inviare richieste a:
Massimo Bilancia - Via P. Amedeo, 49 - 70100 Bari - Tel. 080/238443.

Compro programmi per CBM 64 listati o cassette di giochi utility ecc. Vendo "Alla scoperta dello ZX Spectrum" nuovo mai usato L. 15.000 (anzichè L. 22.000!). Inviare lista programmi a:
Nicola Franchetti - P.zza Libertà, 5 - 35036 Montegrotto Terme (PD) - Tel. 049/794783.

Cerco amici possessori CBM 64 provincia Vicenza per scambio idee, esperienze e programmi. Ho già dei contatti.
Murizio Baratella - Via S. Francesco, 11 - Cas. Post. 139 - 36061 Bassano del Grappa (VI) - Tel. 0424/25888.

Vendo VIC 20 + registratore Commodore + super expander + monitor linguaggio macchina + cartridge gioco scacchi + joystick + molti programmi e libri bellissimi; in blocco a L. 450.000.
Amedeo Fasano - Res. Sagittario - 20090 Segrate (MI) - Tel. 02/2130331.

Vendo per Commodore 64 cartridge hesware (Maze Master) a L. 50.000. Nuova, ancora imballata. Listino L. 75.000.
Giov. Battista Fornari - Via Villoresi, 8 - 20099 Sesto S. Giovanni (MI) - Tel. 02/2474960.

Cambio/vendo programmi per il CBM 64 o 20, inviatemi la vostra lista. Ho molti programmi americani e stranieri. Vendo Jammin, Jet, Pack, Magic miner per il VIC.
Roberto Oselladore - Via Passo S. Boldo, 35/2 - 30030 Favaro Veneto (Venezia) - Tel. 631106.

Compro espansione per alta risoluzione 3 kbyte - VC1211N da chi vuol disfarsene ad un prezzo non superiore a L. 50.000. Chiunque fosse interessato mi telefoni.
Mario Calò - Via Degli Svevi, 49 - 75100 Matera - Tel. 219700 (ore 14,30-16,00).

Vendo cartuccia per Commodore 64 "Sea wolf". Richiedere l'uso del paddle. Prezzo trattabile. Telefonare!
Alessandro Mazzoli - Via Cavour, 29 - 43100 Parma - Tel. 35365.

Dispongo di numerosi programmi per Commodore 64, originali americani, su nastro e su disco. Molti di questi in L.M.
Roberto Onesto - Via G. Pascoli, 31/B - 30038 Spinea (VE) - Tel. 041/991516.

Vendo/cambio molti programmi per VIC 20.
Luigi Lippiello - Via Nicola Fornelli, 2 - 80132 Napoli - Tel. 081/415601.

PICCOLI ANNUNCI

Cambio programmi per Commodore 64 di ogni tipo. Possiedo gestionali utility giochi. Telefonare e/o mandare lista.
Piero Volpi - Via Anconetana, 6 - 06016 S. Giustino (PG) - Tel. 075/856858.

Vendo programmi di utility e giochi. Dispongo inoltre di manuali inglesi tradotti in italiano.
Marcello Cesi - Via Magliana Nuova, 178 scala B/22 - 00146 Roma - Tel. 06/5266009.

Vendo per VIC 20 giochi su cassetta anche inediti a L. 2.500 cad. più costo della cassetta.
Fantone Michele - Via Privata Assarotti - 16043 Chiavari - Tel. 0185/305195 (ore pasti).

Vendo computer VIC 20; un mese di vita. Perfette condizioni. Regalo a chi lo acquista (oltre a cavetti vari) registratore L2N + 2 libri di Basic + 2 cassette piene di giochi. Prezzo L. 420.000.
Gerardo Ventura - V.le Regina Elena, 82 - 65100 Pescara - Tel. 085/23374.

Scambio programmi, listati, idee per Commodore 64. Vendo a poco prezzo Simon's Basic su cassetta. Mandate le vostre liste.
Maurilio Caracci - Via N. Morello, 48 - 90144 Palermo - Tel. (091) 266011.

Vendo VIC 20 + registratore 2CN + espansione 16 k + manuale d'uso in italiano + interconnessioni varie a L. 450.000.
Antonio Schillaci - Via Noventa, 10 - 36100 Vicenza - Tel. 0444/504718 (ore pasti).

Vendo VIC 20 mai usato causa doppio regalo a L. 190.000.
Nino Ardizzone - Via Arenazze, 18 - 66100 Chieti - Tel. 0871/41877.

Cerco in Toscana possessori CBM 64 per scambio esperienze e software. Cerco anche eventuali Clubs in Versilia o in Toscana. Scrivere a:
Marco Pasquini - Via Dei Mille, 81 - 54036 Marina di Carrara (MS) - Tel. 55038.

Cerco espansione grafica 3 kRAM per VIC 20 a buon prezzo, a chi interessa cedo cartridge draw poker + il resto.
Mario Regina - Via Mazzini, 3 - 58015 Orbetello (GR) - Tel. 0564/867441 (ore pasti).

Vendo CBM 4032 schermo 9" fosfori bianchi con floppy 2031, stampante 4022, registratore CN2 e amplificatore altoparlante e interfaccia per RTTY.
Ferruccio Bassini - Via Casanova, 12/A - 26020 Cavatigozzi (Cremona) - Tel. 59077.

Vendo per VIC 20 interfaccia con registratore. Cerco registratore per VIC 20 originale 2CN. Urgente.
Stefano Guerrini - Via Montone, 11 - 48026 - Russi (Ravenna) - Tel. 0544/581318.

Vendo per CBM 64 Simon's Basic (114 comandi) + manuale a L. 38.000 su cassetta e L. 40.000 su disco. Il noto "Basic 4.0" che aggiunge al CBM 64 i comandi del CBM 8000 a L. 36.000 su cassetta e L. 38.000 su disco (con manuale). Spedizione contrassegno. Per informazioni aggiungere bollo.
Francesco D'angelo - Parco Argo, 15/4 - 81031 Aversa (CE).

Causa passaggio ad unità superiore vendo CBM 64, registratore, 100 programmi su cassetta fra i migliori in commercio a prezzi stracciati.
Domenico Riccardi - Via delle Quercie, 1/B - 61016 Pennabilli - Tel. 0541/918312 (ore 20 in poi).

Cerco possessori CBM64 per scambio idee, programmi, esperienze.
Particolare interesse per la grafica e per i programmi statistici.
Luca Tamburro - Via Mazzini, 67 - 80045 Pompei (NA) - Tel. 081/8631918.

Vendo/cambio giochi e package per VIC 20 inespanso 8-16 k. Sono in possesso di oltre 60 giochi originali americani. Il software è disponibile su cassetta o disco; invio liste a L. 1.200.
Roberto Salerno - Via IV Novembre, 24 - 28068 Romentino (NO) - Tel. 60452.

Vendo VIC 20 + registratore C2N + super expander con 3 k + videogame gorf e scacchi + joystick point master, il tutto in imballo originale (4 mesi di vita) a L. 560.000 trattabili.
Telefonare dalle 19,30 alle 21,30.
Valerio Rosso - Via Natale Gallino, 36A/3 - 16164 Pontedecimo (GE) - Tel. 010/798625.

Vendo/compro/cambio programmi per Commodore 64. Scrivere o telefonare a:
Emilio Di Lello - Via Giotto, 3 - 64026 Roseto Degli Abruzzi (TE) - Tel. 085/8992146.

Vendo Commodore 64 software. Simon's Basic, Forth, Easy, script; tutti con istruzioni ed inoltre i più bei giochi e arcade games: Golden baton, Aztec tomb, Benji, Pac-man, Calcio, scacchi, Krazy kong, ecc.
Marco Ellena - Str. S. Vincenzo, 40/35 - 10121 Torino - Tel. 011/8610293.

Vendo/cambio centinaia di programmi per C64. Ho di tutto: utility, videogames, linguaggi, compilatori, backup ecc. Chiedere la lista.
Paolo Vergoni - via Appia, 69 - 06100 Perugia - Tel. 075/66918 (ore 14-15).

Cambio programmi vari su disco o su nastro CBM 64. Scrivetemi.
Augusto Guidotti - Via Lilibeo, 2 - 00100 Roma.

Vendo programmi per VIC 20 (circa 500 di cui 90 in L.M.) a prezzi bassissimi. Cambio sempre per VIC solo programmi in L.M. Cambio anche programmi per lo Spectrum (circa 70). Allegare francobollo per le liste del VIC, gratuite se inviate le vostre.
Giuseppe Mascali - Via R. Margherita, 573 - 98028 S. Teresa Riva (ME) - Tel. 0942/791692.

Compro per VIC 20 espansione di memoria 16 k.
Mario Cavalli - Via Onorato Vigliani, 11/9 - 10100 Torino - Tel. 011/6191596 (dopo ore 18).

Vendo giochi originali inglesi per Commodore 64 in linguaggio macchina, utilities, software di ogni genere. Richiedete la lista completa gratuita.
Maurilio Caracci - Via N. Morello, 48 - 90144 Palermo - Tel. (091) 266011.

Offro vantaggiosi scambi software ad utenti VIC 64. Scrivete inviando vostri elenchi o cataloghi a:
Giulio Cherubini - Via Filippini, 47 - 89100 Reggio Calabria - Tel. 330839.

Vendo/scambio giochi per VIC 20 senza espansione. Inviatemi le vostre liste. I giochi che sono in vendita da parte mia costano pochissimo e sono facilmente sdoppiabili.
Davide Tiozzi - Via N. Sauro, 134 - 48015 Cervia (RA) - Tel. 971328 (ore pasti).

Sinclair

Vendo oltre 300 programmi Spectrum 16/48 k di cui alcuni inediti in Italia, singolarmente o in blocco.
Cosimo Cosmano - Via G. Petraglione, 7 - 70124 Bari - Tel. 080/412111.

Scambio software per ZX Spectrum ZX 16 o 48 k o vendo a L. 5.000 cad. Inviare lista o telefonare.
Alberto Stoppani - Via Roberto Aroigò, 20 - 35126 Padova - Tel. 049/754113.

Vendo/compro programmi per ZX Spectrum 48 k di tutti i tipi. Se siete interessati richiedete o mandate la lista a:
Davide Ardizzola - Via Steria, 100 - 18013 Cervo Ligure.

Vendo causa passaggio ad altro sistema ZX81 1 k come nuovo + 1 manuale in inglese e 1 in italiano ambedue originali + libro dei 66 programmi per ZX 81 tutto a sole L. 130.000 trattabili.
Roberto Mongatti - V.le I° maggio, 12 - 50031 Barberino Mugello (FI) - Tel. 841493.

Vendo ZX81 + 32 k + cavetti per TV e reg. + 2 cassette con più di 100 programmi + manuale in italiano + trasformatore orig. + manuale inglese + imballi originali (luglio '83). Il tutto a L. 225.000 (garanzia in bianco).
Mauro Lorini - Via Botticelli, 44 - 03043 Cassino (FR) - Tel. 0776/481211.

Spectrum area Venezia lezioni individuali software applicativo come programmare; programmi L.M.; come duplicare qualsiasi programma. A richiesta lezioni in tedesco o inglese.
Francesco Vianello/Finzi - Via Cannaregio, 506 - 30123 Venezia - Tel. 041/713354.

Compro per ZX Spectrum 16 k giochi a cassetta a partire da L. 3.000 fino a L. 9.000. Possibilmente gioco Calcio o Pac-man.
Andrea Ferdeghini - Via Del Cantiere - 28053 Castelletto Ticino (NO) - Tel. 0331/920251.

Scambio software su listato per ZX 81 espanso e non, utility e vari articoli, posseggo fotocopie di hardware. Chiedere lista allegando L. 400 in francobolli e specificando cosa interessa.
Fabrizio Martano - Via Don Luigi Sturzo, 7 - 58100 Grosseto - Tel. 0564/492806.

Vendo cassetta con 11 videogiochi fra cui: simulazione di volo, frazioni, statistica, rally e altri 7 programmi ancora più belli, tutto su una cassetta per lo ZX81. Il tutto a L. 40.000.
Andrea Sabatini - Via Cadlolo, 32 - 00136 Roma - Tel. 06/3495086.

Scambio software per ZX Spectrum ampia disponibilità di giochi e utility 16 e 48 k RAM.
Stefano Pezzuto - Via Gabriele D'Annunzio, 74 - 73100 Lecce - Tel. 0832/42441.

Vendo causa doppio regalo ZX Spectrum ancora in garanzia con alimentatore + cassetta dimostrativa + manuale in italiano + volume 77 programmi + 2 cassette gioco (intruders e games 4) tutto a L. 300.000.
Roberto Forestello - Via Visconti, 30 - 20093 Cologno Monzese (MI) - Tel. 2548785.

Scambio programmi, utility e varie per ZX81. Inviare L. 400 in bolli specificando cosa interessa. Possiedo materiale anche per i più diffusi computer.
Fabrizio Martano - Via Don L. Sturzo, 7 - 58100 Grosseto - Tel. 0564/492806.

Scambio programmi per ZX Spectrum, giochi ed utility. Chiedere lista o telefonare. Riprodotti direttamente dal computer. Massima serietà.
Sandro Menegatti - Via Lucera, 13 - 70124 Bari - Tel. 080/223121.

Vendo 3 cassette di programmi per ZX 81: Mazogs, 3-D Defender, Star trek. Tutte da 16 kRAM in L.M. a L. 11.000 cad., usate una sola volta.
Per informazioni scrivere o telefonare a:
Ernesto M. Faccendo - Via Fresine Vasciotte, 17 - 03013 Ferentino (FR) - Tel. 0775/394670 (pomeriggio).

Spectrum 48 k vi sfida a push over, gioco di strategia su scacchiera. Spectrum può anche solo arbitrare. Programma su cassetta L. 20.000 comprese le spese di spedizione. Inviare assegno o vaglia postale.
Anacleto Furlan - V.le Papa Giovanni XXIII, 35/10 - 31015 Conegliano (TV).

PICCOLI ANNUNCI

Cerco programmi per ZX81 16 k, su cassetta. Gestionali, matematici, utility, archivi, contabili, statistici. Inviare listino con prezzi. Anche in inglese.
Giuseppe Soriente - Via Portaromana, 100 - 84015 Nocera Superiore (Salerno).

Cambio/vendo/compro programmi per ZX Spectrum 48 k. Scrivetemi per ricevere liste con programmi a bassissimo costo.
Marisa Banchi - Via Di Tiglio, 15 S. Filippo - 55100 Lucca - Tel. 0583/950688.

Riparo ZX Spectrum non acquistati in Italia, purchè non manomessi, a L. 85.000 fisse. Vendo oric 48 k + cassetta forth + 20 listati di programmi oric, a L. 380.000.
Vialetto Dante - Via Gorizia, 5 - 21053 Castellanza (VA) - Tel. 0331/500713.

Vendo per Spectrum 48 k programma su cassetta che risolve qualunque triangolo, a L. 15.000 comprese spese spedizione. Inviare l'importo in assegno o vaglia postale.
Anacleto Furlan - V.le Papa Giovanni XXIII, 35/10 - 31015 Conegliano (TV).

Vendo/cambio programmi per ZX Spectrum 16/48 k su cassette a prezzi bassissimi, in particolare giochi arcade di ottima qualità. Invio lista a richiesta.
Patrizia Pisani - Via Angelo Emo, 71 - 00136 Roma - Tel. 06/6372386.

Vendo Sinclair ZX 81 + RAM 16 k solamente a L. 16.000. Regalo software comprendente toolkits per Basic e L.M., progr. L.M. per tape files, compilatore Basic, flight simulation ecc.
Giovanni Gugliantini - Via Niccolini, 18 - 20154 Milano - Tel. 02/3495627.

Spectrum soft bank, l'iscrizione è gratuita ed aperta a tutti. Libero accesso per i soci a numerosi programmi e pubblicazioni per lo Spectrum. Per maggiori informazioni scrivere o telefonare presso:
Marino Marinanza - Via Rastrelli, 102 - 00128 Roma - Tel. 06/5203292.

È nato il "Bug Club". Scambiamo e/o vendiamo (5000 max/progr.) software, libri, hardware, ecc. per lo Spectrum. Per informazioni scrivere, allegando bollo, a:
Giovanni Baiano - Via Battisti, 11 Tr. priv. - 80059 Torre del Greco (NA) - Tel. 081/8816572.

Texas

Vendo TI99/4A completo collegamenti TV color e registratore + registratore della Texas Instruments Program Recorder, come nuovo imballo originale a L. 400.000 per passaggio altro sistema.
Giuseppe Solari - Via Caselle Torinese, 52 - 00166 Roma - Tel. 06/6961555.

Vendo TI99/4A + 95 Sparsec + modulo mini memory + editor assembler + manuali + joystick con imballaggi, il tutto con 4 mesi di vita a L. 450.000.
Telefonare allo 011/9835076 (ore pasti).

Vendo per TI 99/4A programmi (giochi, matematica, musica, elettronica) alcuni originali Texas a L. 50.000.
Enzo Santangelo - Via Carnia, 14 - 20132 Milano - Tel. 02/2856924.

Cerco per Texas TI99/4A programmi archivio, grafici, non giochi.
Marco Milletti - Via S. Petronio Vecchio, 42/3 - 40100 Bologna - Tel. 051/398214.

Vendo per TI99/4A circa 80 programmi comprendenti: giochi, matematica, statistica, finanza, commerciali, varie. Vendo anche in blocco. Prezzi irrisori. Spedire francobollo per ricevere elenco e prezzi.
Massimo Boccia - V.le Colli Aminei, 475 - 80131 Napoli - Tel. 081/7434630.

Hai il Texas TI99/4A? E allora cosa aspetti a scrivermi o telefonarmi? Cambio o vendo software per tutti i gusti. Ne ho più di 300 tra TI Basic e Basic esteso. Possibilmente zona Napoli.
Bruno Fonzi - Via Kennedy, 425 - 80125 Napoli - Tel. 081/7602693.

Vendo nuovissimo compact computer Texas CC40 4 mesi di vita causa passaggio sistema superiore. L. 400.000 compresi garanzia, manuale ed imballo. Prezzo non trattabile. Preferibilmente zona Milano.
Gabriele Mazzocchi - Via Dante, 9 - 20077 Sordio (MI) - Tel. 02/9810155.

Vendo per TI99/4A programma, TI/Basic in grado di visualizzare l'orbita di qualsiasi cometa in rapporto alle orbite dei pianeti. Solo L. 9.000, su cassetta. Esecuzione < 1 min.
Carlo Ruocco - Via Rocca Tedalda, 95 - 50136 Firenze - Tel. 670778.

Vendo per TI 99/4A programmi favolosi originali. Usati Basic/ext. Basic listati L. 3.000, cassetta C45 con due programmi L. 10.000. Inviare francobollo L. 400 per invio catalogo.
Massimiliano Verga - V.le Papa Giovanni XXIII, 37 - 20091 Bresso (MI) - Tel. 02/6101151.

Cerco editor assembler nuovo o usato per Texas TI 99/4A purchè funzionante e con manuali.
Luciano Boero - Via Delle Piazze, 5/3 - 16167 Genova - Tel. 010/330421.

Vendo/scambio a prezzi irrisori software per TI 99/4A, extended, assembler, giochi, gestionali, ingegneria. Inviare bollo L. 500 per lista di oltre 400 programmi.
Settimio Perlini - Via 21 gennaio, 152 - 61020 Montecchio (PS).

Vendo TI 99/4A + modulo extended basic + modulo mini memory + programmi applicativi didattici e giochi, tutto a L. 400.000.
Mario Vizzotto - Via Garibaldi, 93 - 31046 Oderzo (TV) - Tel. 0422/713527.

Vendo per TI 99/4A cassetta con eccezionali giochi (Keyboard, Dark dank, Master mind, Pling plong, Crazy climber, Treasure, Harves Basic, ecc.) a L. 5.000.
Riccardo Romagnoli - Via Giovanni XXIII, 6 - 12011 Borgo San Dalmazzo (CN) - Tel. 0171/760482.

Cerco cavo per registratore per il TI 99/4A della Texas Instruments.
Massimo Iaculo - Via Cucciarella - 81024 Maddaloni (CE) - Tel. 434554.

Vendo TI 99/4A + sistema a dischi completo (drive + controller + RS232 + exp. 32 k) + extended Basic + TI writer + multiplan + speech sinth. + terminal emulator + mini memory + giochi L. 2.500.000.
Federico Aloisi - Via Spadini, 3 - 00197 Roma - Tel. 06/3600136.

Vendo in perfette condizioni TI 99/4A, sist. espansione, scheda controllo dischi, drive interno, esp. memoria a RAM, TI extended Basic, 20 dischi, software + cassette giochi a L. 1.800.000 trattabili.
Nicola De Rensis - Via Buonarroti, 28 - 51100 Pistoia - Tel. 0573/27197.

Varie

Acquisto/scambio/vendo programmi su disco e cassetta per computer Atari 400-800.
Luigi Servolini - Via La Spezia, 81 - 00182 Roma - Tel. 06/384488-7581219.

Vendo DAI pc 16 colori grafica fino a 256 x 356 punti, 3 generatori di suoni programmabili, cavi, manuali, 6 numeri della rivista DAInamic (solo per il DAI) L. 1.250.000 trattabili.
Mario Agostini - Via Orafi, 2 - 51100 Pistoia - Tel. 0573/25945.

Vendo, causa acquisto Spectrum, pocket C. Casio PB 100 + esp. (OR-1) (1 k RAM) + int. per registratore + mini stampante il tutto a sole L. 300.000 (valore L. 400.000). Il materiale è inoltre in garanzia fino al 1985.
Luigi Cerabolini - via F. Cavallotti, 91 - 27011 Belgioioso (PV) - Tel. 0382/960816.

Centronics mod. 739 stampante, 1 anno di vita, vendo a L. 700.000. possibilità di fatturazione.
Antonio Gallenzi - Via Cellini, 3 - 20129 Milano - Tel. 02/798076 (ore 15-19).

Vendo cassette per videogame Atari ottime condizioni.
Luca Catellani - Via Rovigo, 2 - 37134 Verona - Tel. 045/581318.

Esposimetro digitale PentaxSpotmatic, angolo di lettura 1 grado, cambio con Sinclair Spectrum oppure vendo L. 400.000.
Giovanni Felice Massimo - Via Strada 186 - 67100 L'Aquila - Tel. 0862/313164 (ore 14-15).

Svendo riviste di elettronica di ogni tipo italiane e straniere a prezzi bassissimi. Chiedere elenco dettagliato.
Rinaldo Ricci - Via G. Giusti, 15 - 18038 Sanremo (IM).

Macchina fotografica Reflex automatica e manuale tempi da 8 sec. a 1/1000 + B con flash elettronico e diversi libri di fotografia cambio con stampante oppure con unità a dischetti per VIC 64.
Lorenzo Longagna - Via Piave, 20/7 - 17100 Savona - Tel. 019/25322.

Gruppo programmatori, mette a disposizione la propria esperienza per la realizzazione di software per qualsiasi tipo di impiego su qualunque tipo di sistema. Per informazioni scrivere a:
Massimiliano Piarulli - Via Dei Caduti Partigiani A/33 - 70126 Bari - Tel. 080/338110.

Vendo Atari 2600 con paddle, tre cassette L. 189.000. Atari 800 L. 800.000. Drive 810 L. 650.000.
Eugenio Candi - Via Battiferro, 1 - 40128 Bologna - Tel. 051/368936.

Vendo Atari 800, drive 810, drive 1050, completi di programmi e stampante Centronics 730. Prezzo da convenire.
Ugo Donini - Via Sacco, 1 - 40128 Bologna - Tel. 051/516888.

Vendo consolle porta computer con piano per monitor o TV completo di prese e cavi a sole L. 50.000.
Barca Giuseppe - Via Tre Re, 29 - 20047 Brugherio (MI) - Tel. 039/879211.

L'Italian Bit Club, il maggiore computerclub della Liguria, mette a tua disposizione i suoi oltre 100 soci, 1000 programmi, 200 manuali.
Sasha Drago - Via Zara 5 - 16144 Genova - Tel. 010/313153.

Vendo microprocessor II Apple compatibile 64 k, scheda Pal TV color, modulatore, interfaccia parallela per stampante, più controller, più driver slim line originale L. 1.200.000.
G.C. Giacobbe - Via Finocchiara, 46 - 16144 Genova - Tel. 010/825537.

Vendo giochi per Sharp MZ-80 A, se siete interessati scrivete per richiedere lista a: John Ceresi - Via Mazzini - 18016 S. Bartolomeo Al Mare.

Acquisto-scambio-vendo programmi (utility-games) per computer Atari 400-800. Luigi Servolini - Via La Spezia, 81 - 00182 Roma - Tel. 06/384488 - 7581219.

PICCOLI ANNUNCI
PERSONAL SOFTWARE

Sei un lettore di PERSONAL-SOFTWARE e vuoi entrare in contatto con tutti gli altri lettori per comperare, cambiare o vendere software? Spedisci questo tagliando a Gruppo Editoriale Jackson - Via Rosellini, 12 - 20124 Milano

Nome .... Cognome ....

Via .... C.A.P. ....

Città .... Tel. ....

CEDOLA DI COMMISSIONE LIBRARIA
PERSONAL SOFTWARE

Da inviare a Gruppo Editoriale Jackson - Via Rosellini, 12 - 20124 Milano

Nome Cognome

Indirizzo

Cap. Città Provincia

Partita I.V.A. (indispensabile per le aziende)

Si richiede l'emissione della fattura ☐

Inviatemi i seguenti libri:

Codice Libro	Quantità	Codice Libro	Quantità	Codice Libro	Quantità	Codice Libro	Quantità

☐ Pagherò al postino il prezzo indicato + L. 2.000 per contributo fisso spese di spedizione.

☐ Allego assegno n° .... di L. ....

☐ Non abbonato ☐ Abbonato sconto 20% ☐ l'Elettronica ☐ Elettronica Oggi ☐ Automazione Oggi ☐ Elektor ☐ Informatica Oggi ☐ Computerworld ☐ Bit ☐ Personal Software ☐ Strumenti Musicali ☐ Videogiochi

Data .... Firma ....

UNA PUBBLICAZIONE
DEL GRUPPO EDITORIALE JACKSON

PERSONAL SOFTWARE

ANNO 3 N. 18 MAGGIO 1984

DIRETTORE RESPONSABILE: Giampietro Zanga

CAPO REDATTORE: Lorenzo Barrile

DIRETTORE TECNICO: Riccardo Paolillo

CONSULENZA: Giacomo Bortone

HANNO COLLABORATO A QUESTO NUMERO:
U. Barzaghi, A. Motta, A. Guida, C. Poma, V. Riva, M. Morchio, F.Cerulo, C. Cappelli, S. Borsani, M. Lenzi, M. Spero, M. Cappelletti, M. Spada.

UFFICIO ABBONAMENTI: Tel. 6880951/2/3/4/5

GRAFICA E IMPAGINAZIONE: Luigi Chiesa

USA: G.E.J. Publishing Group, Inc. a Subsidiary of - Gruppo Editoriale Jackson 1143 Quince Avenue - 94087 Sunnyvale (CA) Tel. (408) 7730103 - Telex: 4995972

GRAN BRETAGNA: G.E.J. Publishing L.t.d. 187 Oxford Street London WIR1AJ Tel. (01) 4392931 - Telex: (051) 21248

PUBBLICITA': Concessionario per l'Italia e l'Estero Reina S.r.l. Via Washington, 50 - 20146 Milano Tel. (02) 4988066/7/8/9/060 (5 linee r.a.) Telex: 316213 REINA I

PUBBLICITA' PER USA E CANADA: International Media Marketing 16704 Marquardt Avenue, P.O. Box 1217 Cerritos (CA) 90701 - (213) 926-9552

FOTOCOMPOSIZIONE: LINEACOMP S.r.l. Via Rosellini, 12 - 20124 Milano

STAMPA: Reweba - Brescia

Concessionario esclusivo per la diffusione in Italia e all'Estero: SODIP - Via Zuretti, 25 - 20125 Milano

AUTORIZZAZIONE ALLA PUBBLICAZIONE: Tribunale di Milano n. 69 del 20/2/1982

Spedizione in abbonamento postale Gruppo III/70
Prezzo della rivista L. 3.500. Numero arretrato L. 7.000.
Abbonamento annuo (10 numeri) L. 28.000; per l'Estero L. 44.800
I versamenti vanno indirizzati a: Gruppo Editoriale Jackson Via Rosellini, 12 - 20124 Milano - mediante emissione di assegno bancario, cartolina vaglia o utilizzando il C/C postale numero 11666203.
Per i cambi di indirizzo, indicare, oltre naturalmente al nuovo, anche l'indirizzo precedente, ed allegare alla comunicazione l'importo di L. 500, anche in francobolli.

Il Gruppo Editoriale Jackson è iscritto nel registro Nazionale della stampa al n. 117 - Vol. 2 - Foglio 129 in data 17-8-1982

GRUPPO EDITORIALE JACKSON S.r.l.

DIREZIONE, REDAZIONE, AMMINISTRAZIONE: Via Rosellini, 12 - 20124 Milano - Telefoni: 68.80.951/2/3/4/5

SEDE LEGALE: Via G. Pozzone, 5 - 20121 Milano

DIREZIONE EDITORIALE: Giampietro Zanga e Paolo Reina

COORDINAMENTO EDITORIALE: Daniele Comboni

DIREZIONE AMMINISTRATIVA: Giuliano Di Chiano